Design of a Miniaturized Microstrip Antenna with a Slotted Ground Plane using Trust-Region Optimization and Surrogate Modeling

Final Report

S.J. Kotze
u13026242

Submitted as partial fulfilment of the requirements of Project EPR 400

in the Department of Electrical, Electronic and Computer Engineering

University of Pretoria

Monday 20th January, 2020

Study leader: Prof. J.P. Jacobs

Part 1. Preamble

This report describes work that I have completed for my final year project. This work revolved around the development of software that is capable of miniaturizing a microstrip patch antenna through the use of multivariate optimization and surrogate modelling.

Project proposal and technical documentation
This main report contains a copy of the approved Project Proposal (as Part 2 of the report). Complete technical documentation appears as part of the additional material submitted on the electronic medium that accompanies this printed report.

Project history
This project makes use of existing algorithms on multivariate optimization as well as surrogate modelling. Some of the algorithms I used were adapted from the algorithms found in the following journal articles: [1] and [2] Where other authors' work has been used, it has been cited appropriately, and the rest of the work reported on, is entirely my own.

Language editing
This document has been language edited by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.

My language editor was _____.

_____ _____
Language editor signature *Date*

Declaration
I, _____ understand what plagiarism is and have carefully studied the plagiarism policy of the University. I hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the Internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else.

Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination / evaluation about the work.

I certify that the Project Proposal appearing as the Introduction section of the report is a verbatim copy of the approved Project Proposal.

_____ _____
S.J. Kotze Date

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

TRO	Trust-Region Optimization
PSO	Particle Swarm Optimization
BCO	Bee-Colony Optimization
FDTD	Finite-difference time-domain method
MoM	Method of Moments
RMSD	Root-Mean-Square Deviation
DH	Data Handler
OH	Optimization Handler
CPU	Central Processing Unit
SH	Surrogate Handler
LIH	Lua Interface Handler
ASH	Antenna Simulation Handler
AOH	Antenna Optimization Handler

Part 2. Project Definition: Approved Project Proposal

This section contains the problem identification in the form of the complete and approved Project Proposal, unchanged from the final approved version.

For use by the project lecturer	Approved		Revision Required	

To be completed by the student						
PROJECT PROPOSAL 2019				Project No.	JPJ 10	Revision no.	Rev. 0
Title	Surname	Initials	Student no.	Study leader (title, initials, surname)			
Mr.	Kotze	S.J.	13026242	Dr. J.P. Jacobs			
Project title Design of a miniaturized microstrip antenna with a slotted ground plane using trust-region optimization and surrogate modeling							

Language editor name	Language editor signature
Student declaration I understand what plagiarism is and that I have to complete my project on my own.	**Study leader declaration** This is a clear and unambiguous description of what is required in this project
Student signature	Study leader signature

1. Project description

What is your project about? What does your system have to do? What is the problem to be solved?

This project is about designing a miniaturized microstrip antenna that radiates over a slotted ground plane. This is accomplished through the use of a trust region optimization algorithm that utilizes a surrogate model. The system accepts a microstrip antenna geometry as an input at which point the system approximates the objective function through the use of the surrogate model and an optimization algorithm. The system then attempts to minimize the objective function within a certain region about the initial design point, namely the trust region. This is accomplished by using the optimization algorithm in conjunction with the cost function in order to find the minima within the trust region.. When a local minimum has been found within the trust region a full-wave simulation is run using the microstrip antenna geometry of the trial point. The surrogate model output of the trial point, namely the return loss vs. frequency spectrum, will then be compared to that of the full-wave simulation of the trial point in order to determine if the step taken is accurate. Based on the accuracy of the step the new trial point will then be used and the trust region expanded if the step was accurate, or the step will be recalculated with a smaller trust region. This process repeats iteratively until either the input antenna geometry converges or the trust region becomes too small. The final microstrip antenna geometry will then be printed and tested in real world conditions.

2. Technical challenges in this project

Describe the technical challenges that are beyond those encountered up to the end of the third year and in other final year modules

• 2.1 Primary design challenges

- Designing the surrogate model
- Designing the optimization algorithm
- Designing a suitable cost function for trust region optimization algorithm
- Designing the trust region algorithm

2.2 Primary implementation challenges

- Determining which surrogate model to use, where possible surrogate models that could be used are polynomial response surface, kriging, support vector machines, and artificial neural networks.
- Selection and implementation of the best cost function to accurately determine the quality of a trial point
- Selection and implementation of the best optimization algorithm, where possible optimization algorithms that could be used are particle swarm optimization and differential evolution optimization. The optimization algorithm will be written by the student.

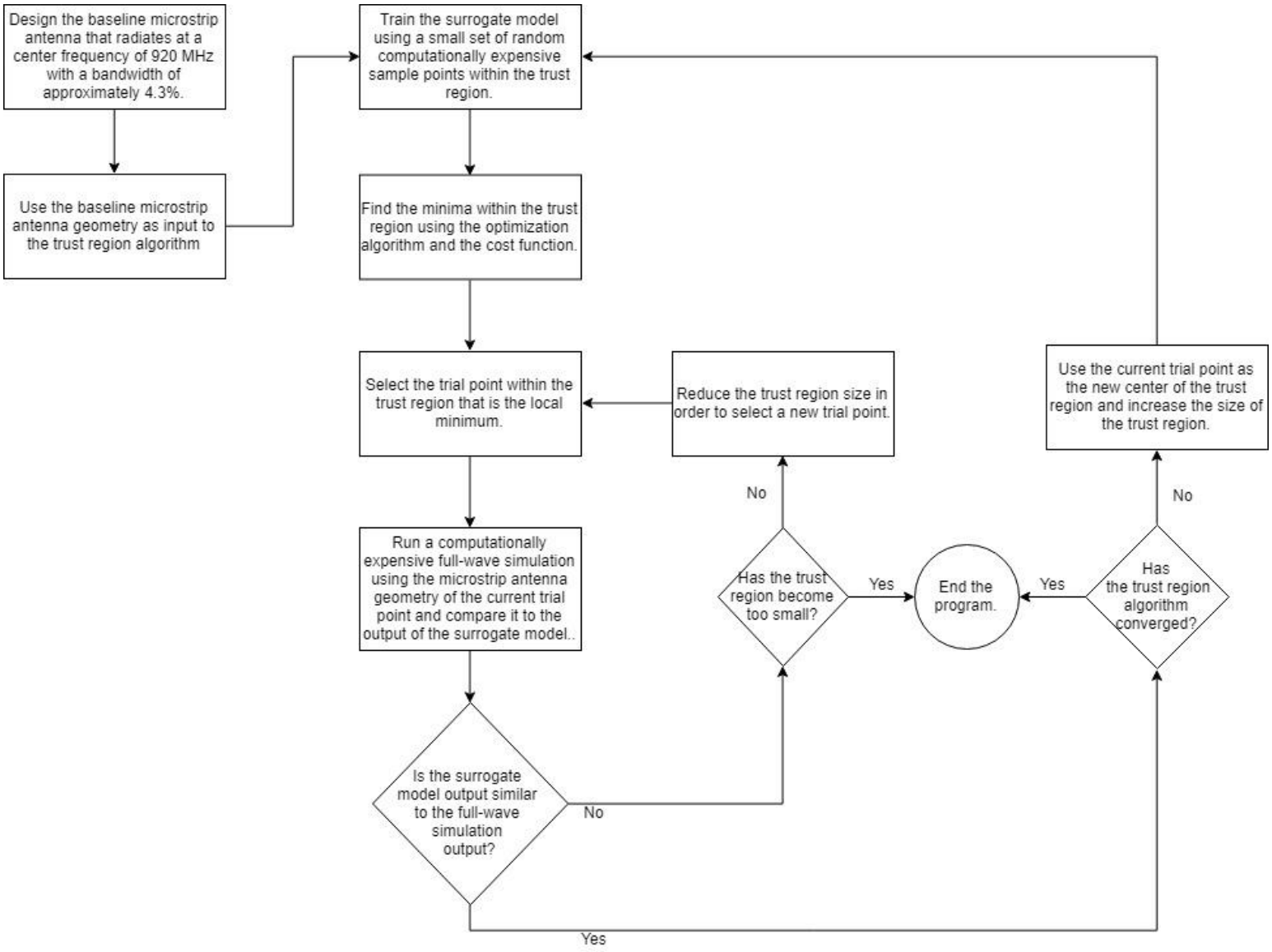
3. Functional analysis

3.1 Functional description

Describe the design in terms of system functions as shown on the functional block diagram in section 3.2. This description should be in narrative format

The first step in the process would be to design a baseline microstrip antenna with a bandwidth of approximately 4.3%. This will allow the microstrip antenna to operate between the frequencies of 900 MHz – 940 MHz, which would allow the microstrip antenna to operate within the GSM-900 band. At this point the trust region optimization algorithm will then be given the aforementioned baseline microstrip antenna geometry as input. The objective function will be approximated by training the surrogate model with a small set of computationally expensive randomly selected sample points within the trust region. An optimization algorithm will then be used in conjunction with the cost function to surface fit the surrogate model, which ideally will allow the optimization algorithm to find the minima within the trust region. It should be noted that the cost function being used could for example be the magnitude of the reflection coefficient at the frequency 920 MHz. A step will then be chosen within the trust region using the local minima found by the optimization algorithm in order to minimize the objective function. In order to then test whether or not the step is accurate a computationally expensive simulation will be run using the trial point from the previous step and its return loss vs. frequency spectrum will be compared to that of the surrogate model at the trial point. If the step is accurate then the trial point is used as the center of the trust region in the next iteration of the algorithm and the trust region is expanded, however if the step was not accurate then the step will be recalculated with a smaller trust region. This process then repeats until either the algorithm has converged and no more steps are being taken due to a minimum having been reached or until the trust region becomes too small. Once the final antenna geometry has been obtained it will be printed and tested in real world conditions.

3.2 Functional Block Diagram



4. System Requirements and Specifications

These are the core requirements of the system or product (the mission-critical requirements) summarized in table format.

	Requirement 1: Fundamental Functional and Performance Requirement	**Requirement 2**	**Requirement 3**
1. Core mission requirements of the system or product. Solution of the problem will be the most important requirement. Capture this in the set of requirements.	A miniaturized microstrip antenna must be designed through the use of a trust region algorithm that radiates over a slotted ground plane in the frequency range of 900 MHz to 940 MHz where that frequency range would be the -10 dB bandwidth of the microstrip antenna.	A miniaturized microstrip antenna must be designed through the use of a trust region algorithm that radiates over a slotted ground plane and has been miniaturized at least 45% compared to the baseline microstrip antenna.	A miniaturized microstrip antenna must be designed through the use of a trust region algorithm that radiates over a slotted ground plane and has a broadside gain of at least 1.25 dBi.
2. What is the target specification (in measurable terms) to be met in order to achieve this requirement?	The miniaturized microstrip antenna designed through the use of a trust region algorithm must operate at a bandwidth of approximately 4.3% with an error of no more than 5% on the band edges.	When compared to the baseline microstrip antenna designed to operate in the frequency range of 900 MHz to 940 MHz the total surface area of the miniaturized microstrip antenna must be reduced by at least 45%.	The miniaturized microstrip antenna designed through the use of a trust region algorithm must have a gain no less than 1.25 dBi.
3. Motivation: how will meeting this specification solve the problem?	The 900 MHz to 940 MHz frequency band is a very commonly used GSM frequency band in mobile cellular devices, thus it would be ideal if the time spent designing the microstrip antenna can be minimized through automatic optimization through the use of a trust region algorithm.	Since the GSM-900 frequency band is commonly used in mobile cellular devices, which are getting smaller all the time, keeping the total surface area used by the microstrip antenna of the device to a minimum allows manufacturers to save money and perhaps use more components in a product.	If the gain of the antenna is too small then the antenna functionality would decrease as a possible end user might have to be much closer to a signal source than would be necessary with other similar devices.
4. How will you demonstrate at the examination that this requirement has been met?	The frequency spectrum of the miniaturized microstrip antenna will be tested through the use of a network analyzer.	The total surface area of the miniaturized microstrip antenna can be measured physically and compared to the design of the initial microstrip antenna.	The gain of the miniaturized microstrip antenna will be tested by measuring the radiation intensity of the miniaturized microstrip antenna through the use of a receiving antenna .
5. What is the deliverable? What are the aspects that you will design and implement yourself to meet this requirement? If none, indicate clearly.	The deliverable is the miniaturized microstrip antenna and the aspects to be designed are the frequency band that was chosen for the microstrip antenna to radiate over.	The deliverable is the miniaturized microstrip antenna. The miniaturization will not be directly designed but will be automatically designed through the use of the trust region optimization algorithm.	The deliverable is the miniaturized microstrip antenna. The gain will not be directly designed but will be automatically designed through the use of the trust region optimization algorithm.
6. What are the aspects to be taken off the shelf to meet this requirement? If none, indicate clearly.	Once the miniaturized microstrip antenna design has been completed by the trust region algorithm the antenna will be printed by an external third party.	Once the miniaturized microstrip antenna design has been completed by the trust region algorithm the antenna will be printed by an external third party.	Once the miniaturized microstrip antenna design has been completed by the trust region algorithm the antenna will be printed by an external third party.

4. System Requirements and Specifications (continued)

These are the core requirements of the system or product (the mission-critical requirements) summarized in table format.

	Requirement 4	Requirement 5	Requirement 6
1. Core mission requirements of the system or product. Solution of the problem will be the most important requirement. Capture this in the set of requirements.	A trust region optimization algorithm that is capable of automatically optimizing an input microstrip antenna geometry within three hours.	A trust region optimization algorithm that uses no more than 45 full-wave simulations in order to optimize a given microstrip antenna geometry	
2. What is the target specification (in measurable terms) to be met in order to achieve this requirement?	Once the trust region optimization algorithm has been given the input microstrip antenna geometry the algorithm must be capable of miniaturizing the input geometry within three hours whilst keeping the frequency bandwidth within in the desired band and whilst maintaining a gain of no less than 1.25 dBi.	The amount of full-wave simulations utilized by the algorithm must be no more than 45 simulations.	
3. Motivation: how will meeting this specification solve the problem?	By meeting this specification the trust region algorithm will allow antenna designers to reduce the time spent designing an antenna which will allow them to more easily complete feasibility and sensitivity studies.	If the algorithm utilizes to many full-wave simulations then it will become time consuming and thus not be fast enough.	
4. How will you demonstrate at the examination that this requirement has been met?	The algorithm can be run during the exam to demonstrate the amount of time it takes for it to complete.	The amount of full-wave simulations can be measured during the execution of the program	
5. What is the deliverable? What are the aspects that you will design and implement yourself to meet this requirement? If none, indicate clearly.	The deliverable is the trust region optimization algorithm that utilizes surrogate modeling. The aspects to be designed and implemented are the surrogate model, the optimization algorithm, the cost function, and the trust region algorithm.	The deliverable is the trust region optimization algorithm that utilizes surrogate modeling. The aspects to be designed and implemented are the surrogate model and the cost function.	
6. What are the aspects to be taken off the shelf to meet this requirement? If none, indicate clearly.	N/A	N/A	

5. Field conditions

These are the core requirements of the system or product (the mission-critical requirements) summaries in table format.

	Field Condition 1	Field Condition 2	Field Condition 3
Field condition requirement. In which field conditions does the system have to operate. Indicate the one, two, or three most important field conditions.	The antenna should be within receiving range of the transmitter in an indoor environment.		
Field condition specification. What is the specification (in measurable terms) for this field condition?	The antenna should be within 40 meters of the transmitter in an indoor environment.		

6. Student tasks

6.1 Design and implementation tasks

List your primary design and implementation tasks in bullet list format (5-10 bullets). These are not product requirements, but your tasks

- Design of a baseline microstrip antenna that operates in the 900 MHz to 940 MHz frequency band.
- Selection and implementation of a surrogate model that is both fast and reliable.
- Design and implementation of a suitable cost function.
- Selection and implementation of an appropriate optimization algorithm.
- Design and implementation of the trust region algorithm.

6.2 New knowledge to be acquired

Describe what the theoretical foundation to the project is, and which new knowledge you will acquire (beyond that covered in any other undergraduate modules)

- Antenna design and parameters.
- Surrogate modelling.
- Optimization algorithms
- Trust-region optimization methods.

Part 3. Main Report

1. Literature Study

Microstrip antennas, also referred to as patch antennas or planar antennas, are radio antennas that usually have a surface area anywhere in the range of 100cm x 100cm to 1cm x 1cm or smaller . The surface area of a microstrip antenna is highly dependent on its operating frequency. Microstrip antennas then consist of, but are not limited to, the following integral components:

- A feed
- A flat ground plane
- A substrate
- A flat radiating plane

The substrate of the microstrip antenna lies atop the ground plane and can be one of several different materials. The radiating plane and the feed are then positioned on top of the substrate with the feed connected to the radiating plane and extending outwards to the end of the substrate. The radiating plane and the feed are commonly parallel to the ground plane with the exception of extreme cases. The construction of microstrip antennas is commonly achieved through the same process that is used to print pcb circuits. Some common substrates along with their properties are as follows:

The common microstrip antenna geometry would then look as follows:

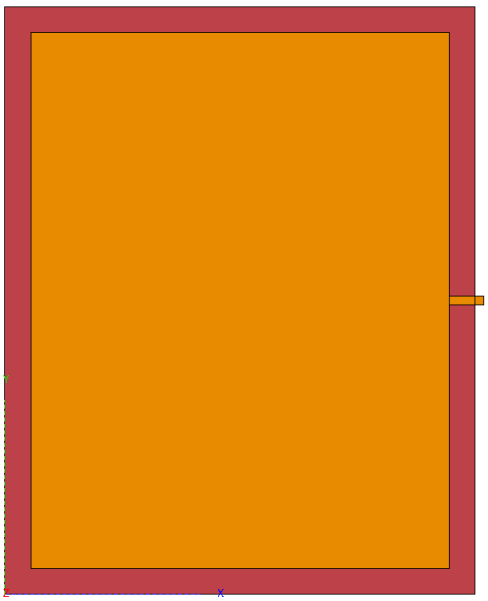


Figure 1.
Common Microstrip Antenna Geometry Radiating Plane Viewed From Above

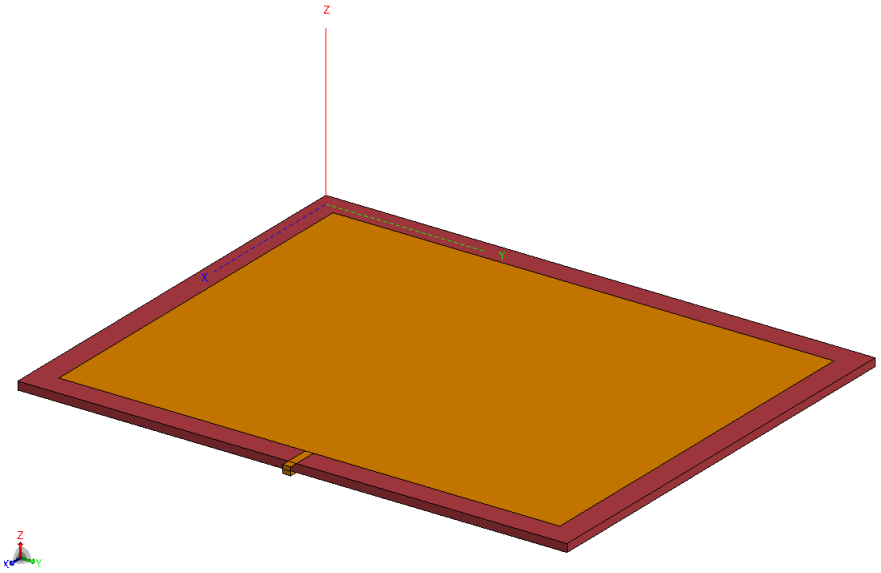


Figure 2.
Common Microstrip Antenna Geometry Radiating Plane

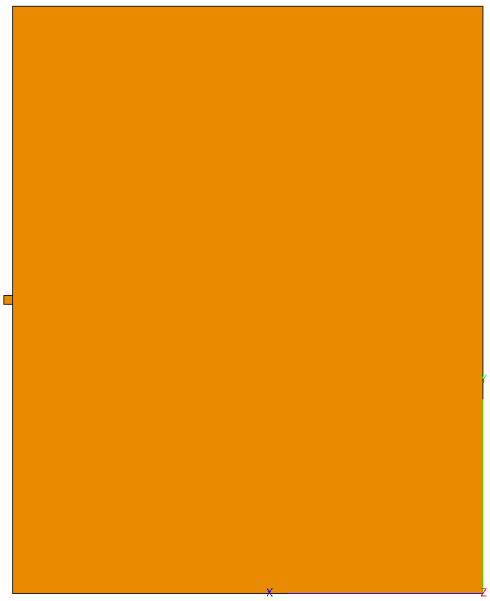


Figure 3.
Common Microstrip Antenna Geometry Ground Plane Viewed From Below

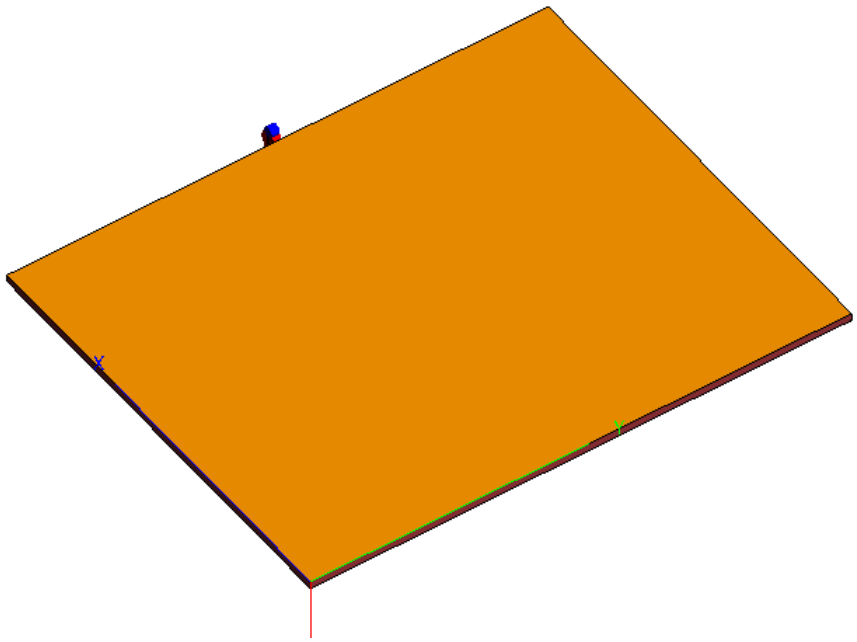


Figure 4.
Common Microstrip Antenna Geometry Ground Plane

It can then be seen in Figure 1 to Figure 4 that the substrate is the crimson colored cuboid with the radiating plane, feed, and ground plane being the dark orange coloured surfaces.

Some of the common advantages of microstrip antennas are as follows:

- Microstrip antennas are typically very low profile
- The ground plane, radiating plane, and feed can be easily etched onto the substrate through the use of modern pcb printing techniques
- The manufacturing cost of microstrip antennas are lower than that of non-microstrip antennas
- Microstrip antennas can operate at multiple frequency bands [3] [4]
- Microstrip antennas are light weight

With the disadvantages of microstrip antennas being:

- Due to dielectric and conductor losses the efficiency of microstrip antennas are usually low
- Due to the small size of microstrip antennas they tend to have lower gain than non-microstrip antennas
- Microstrip antennas tend to have higher levels of cross polarization radiation

- Microstrip antennas tend to have lower power handling capabilities
- Microstrip antennas tend to have narrower impedance bandwidths

Keeping the advantages of microstrip antennas in mind it is clear to say that due to their low profile and versatile operating frequencies, they are clearly the most suitable candidate with regards to the use of antennas in electronics. Some of the more common applications of microstrip antennas in electronic devices are:

- Antennas for wireless cellular phones
- Antennas for IoT devices
- Antennas used in laptop computers

Now suppose that we can create a new microstrip antenna that operates at the same frequency as its predecessor but the new microstrip antenna is smaller and therefore requires less space in an electronic device. If the manufacturer of an electronic device were to replace the predecessor microstrip antenna with the new microstrip antenna they would be faced with two options:

1. Reduce the size of the electronic device and therefore likely the production cost of the electronic device
2. Use the space that is no longer being used by the microstrip antenna to add additional electronic components that would increase the market value of the electronic device
3. Use the space that is no longer being used by the microstrip antenna by replacing some of the other components in the electronic device with better components, thus increasing the market value of the electronic device

It is clear then to see that not only can a manufacturer easily increase their profits should they use a miniaturized version of a microstrip antenna.The underlying problem then becomes a matter of how to miniaturize the microstrip antenna. Some of the more common and prominent methods of miniaturizing microstrip antennas are as follows:

- Array antennas
- Alter the substrate size
- Impedance matching through introduction of cavities
- User of superstrates
- Introduction of slots to the radiating plane
- Introduction of slots to the ground plane

Of the above options, array antennas and better substrates cost more money. Adding slots to the radiating and ground planes can only reduce the cost of the antenna since adding a slot requires no additional parts and reduces the amount of material required to print the conducting planes.

This project then intends to focus on the miniaturization of a microstrip antenna by creating a slotted ground plane. In a broad sense, adding a slot to the ground plane changes the impedance of that ground plane when looking at the frequency spectrum. In other words, adding a slot to the ground plane shifts the operating frequency of the microstrip antenna. If we then take into consideration the fact that reducing the size of the microstrip antenna also shifts the frequency spectrum then it becomes clear that in order to miniaturize a microstrip antenna by adding slots, it is necessary for both the shift in frequency due to the miniaturization and the shift in frequency due to the slot addition to cancel each other out. However, this is easier said than done due to the fact that changing the dimensions of a slot as well as the position of the slot with respect to the plane can drastically change the operating frequency of a microstrip antenna.

The problem to be solved by this project is then the automatic miniaturization of a microstrip antenna via creation of a slotted ground plane with a focus on the placement of the slot as well as the shape and dimensions of the slot. In order to create this slotted ground plane the following methods can be used:

- Use of common slot placements that have been developed for the purpose of microstrip antenna miniaturization such as
- Optimization through spacial mapping

Of the above slot creation techniques, this project will focus on optimization through spacial mapping. Spacial mapping of a multivariate sample space can be accomplished through the use of one of several multivariate optimization. Some of the common multivariate optimization algorithms used to optimize microstrip antennas are as follows:

- PSO
- TRO
- BCO
- NM
- Simmulated Annealing

The general concept, in terms of functionality, for nearly all multivariate optimization algorithms is as follows:

1. The algorithm is provided with an objective function and a data set
2. The algorithm then creates and / or moves a set of candidates whose positions within the objective function were determined either numerically or randomly

3. The algorithm is then likely to determine values of all the candidates that have either moved or been created. Once this is complete the algorithm saves the candidate with the most desirable outputs
4. Return to Step 2 until one of the algorithm's exit requirements is met
5. After the algorithm's exit requirements have been met the algorithm returns the point within the objective function that yielded the most desirable output values

In order to implement a multivariate optimization algorithm we then require two final parts, namely:

- An objective function
- A means of generating a useful data set with which the objective function can be mapped

It should be noted at this point that in Step 2 of the explanation of the functionality of multivariate optimization algorithms it is stated that the algorithm creates and / or moves a set of candidates within the dimensions of the objective function. With this in mind it can be noted that through the appropriate selection of one of the multivariate optimization algorithms, the creation of a useful data set becomes possible. To this end this report focuses on the use of the Trust-Region Optimization (TRO) algorithm to create candidate microstrip antenna geometries within the objective function. The TRO algorithm has the added functionality of performing well on non-linear problems and is therefore a very suitable candidate for the proposed problem.

The final requirement to be met in order to obtain a working solution is then the creation of an objective function. In the field of engineering the practice of creating an objective function is commonly accomplished through the use of a surrogate model. A surrogate model is a essentially a black box that is used to map the inputs of a data set to the outputs of that data set. Some of the more common surrogate modelling techniques are as follows:

- Artificial Neural Network (ANN)
- Support Vector Machine Regression (SVMr)
- Kriging

This project then focuses on the miniaturization a microstrip antenna through the creation of a slotted ground plane. The slotted ground plane is created through the use of Trust-Region Optimization accompanied with surrogate modelling.

Some previous works that focused on microstrip antenna miniaturization through similar methods are as follows:

1.1 Design of a Band-Notched UWB Microstrip Antenna with an Inverted-L Slotted Ground Plane

In this paper [5] an Ultra-Wide Band (UWB) microstrip antenna was designed through the use of an slotted ground plane in the shape of an inverted-L. The designed microstrip antenna exhibited band-notched characteristics that could be easily controlled through the use of the inverted-L shaped slots that were introduced to the ground plane.

After the design and implementation of the microstrip antenna the results showed that through the introduction of the slots to the ground plane a band-notch was introduced at a frequency of 5.2 GHz. With the exclusion of the band-notch characteristic of the microstrip antenna the final implementation final microstrip antenna had a VSWR of less than 2.0 within the 2.8 GHz to 10.7 GHz frequency band with a rejection band within the 4.8 GHz to 5.7 GHz frequency band.

1.2 Calculation of Optimized Parameters for a Rectangular Microstrip Patch Antenna using Particle Swarm Optimization

In this paper [6] a multivariate optimization algorithm, PSO, was used to optimize the parameters of a rectangular microstrip antenna. The parameters of the microstrip antenna that were optimized via PSO are parameters such as the length and the width of the microstrip antenna. In other words, given the desired resonant frequency as well as the parameters of the substrate the PSO algorithm was used to calculate the length and the width [7] of standard microstrip antenna geometry. This was accomplished through the use of only the PSO algorithm with an average execution time of approximately 2.7 seconds, using 50 iterations of the algorithm.

This paper then goes on to compare the results achieved through the use the PSO algorithm to results that were previously achieved through the use of a Genetic Algorithm (GA). The comparison of the results proved that a relatively simple mathematical problem could quickly and accurately be solved through the use of the PSO algorithm; more so than through the use of a GA.

1.3 Design and Optimization of Dual Band Microstrip Antenna Using Particle Swarm Optimization Technique

In this paper [3] a dual band microstrip antenna was designed and optimized through the use of the PSO algorithm. The parameters of the microstrip antenna that were focused on were parameters that relate to multiple rectangular slots on the radiating plane of the microstrip antenna. The effects of these slots then produce a frequency response that is optimized at two distinct frequencies whilst maintaining the same polarization for each of the frequencies. The rectangular slots were designed in order for the $TM0_1$ and $TM0_3$ modes to produce as large a frequency range as possible. Another parameter that was optimized in order to reach an optimum impedance matching at both the desired frequencies was the position of the feed, which is due to the fact that the feed has a very large impact on the frequency response of a microstrip antenna.For such a design, the feed position is required to produce optimum impedance matching

at both frequencies.

The results of the paper concluded that through the correct selection of slot dimensions and positions it is possible to optimize and therefore miniaturize a dual band microstrip antenna via the PSO algorithm. The flexibility of the fitness function used during the process played an integral role in the success of the design and implementation of the desired microstrip antenna. It wass therefore determined that the PSO algorithm is a suitable algorithm with regards to antenna optimization problems which are more often thatn not non-linear and multi-modal.

1.4 Fast Simulation-Driven Feature-Based Design Optimization of a Compact Dual-Band Microstrip Antenna with a Branch-Line Coupler

In this work [4] a dual-band microstrip antenna with a branch-line coupler was optimized through the use of surrogate modelling. The surrogate model performed efficient feature-based optimization of a baseline microstrip antenna without the use of derivatives whilst utilizing the concept of response features. In this paper the fact that the response of variations in the branch-line coupler were more linear than the several other parameters of the microstrip antenna geometry were exploited in an effort to make the surrogate modelling more accurate. The surrogate model was then embedded within a trust-region framework in order to ensure that the surrogate model remained accurate.

The resulting dual-band microstrip antenna was simulated using only 24 high-fidelity Electro-Magnetic (EM) simulations branch-line coupler. Due to the the fact that the choice of parameters exhibiting nearly linear responses the process of optimizing the microstrip antenna was computationally inexpensive whilst being conducted through the use of high-fidelity EM simulations.

1.4.1 Cost-Efficient Design Optimization of Compact Microstrip Antennas With Improved Bandwidth

In this paper [8] compact microstrip antennas were optimized through the use of a fast design and optimization process that was assisted by surrogate modelling. Due to the discrepancies exhibited between varying fidelities used in EM simulations frequency scaling assisted by response correction techniques for residual model misalignment were used as model correction tools. In order to then further reduce the computational cost of the design and optimization process the optimization process was designed in such a manner that it could adaptively control the number of parameters being optimized.

The results then showed that the final microstrip antenna had bandwidth of 29% within the frequency range of 4.5 GHz to 6.0 GHz where the original microstrip antenna had a bandwidth of approximately 9% at the same center frequency. The optimized antenna then had a surface area of $645mm^2$ which is 25% reduction in area compared to the baseline microstrip antenna.

2. Approach

2.1 Problem Breakdown

The objectives for this project that directly involve the miniaturized microstrip antenna pertain to the amount of miniaturization, the gain and the frequency of the miniaturized microstrip antenna. In order to meet these objectives a software solution is required that can automatically optimize a baseline microstrip antenna that operates at a user defined frequency. The underlying problems are then broken down into smaller, individual problems that are capable of being implemented and tested separately. The following sub-problems were then identified within the system:

- Random microstrip antenna geometry creation
- Microstrip antenna geometry simulation
- Microstrip antenna geometry fitness evaluation
- Microstrip antenna geometry decomposition into a data set
- Surrogate training
- Surrogate mapping via a multivariate optimization algorithm
- Trust-Region Optimization (TRO)

It should be noted that in order for these sub-problems to collectively form a fully functional software solution they would need to be completely flexible. This is in part due to the fact that the microstrip antenna geometry creation is a random process.

The first conceptual design for the functionality of the software solution can then be found in Figure 5 at the end of this Section.

2.2 Design Alternatives

With regards to design alternatives within the implementation of the software solution the following aspects must be considered:

- The simulation software
- The surrogate model
- The method of data acquisition

2.2.1 Simulation Software

Multiple third-party software applications exist that are capable of modelling and simulating microstrip antenna geometries. Some of the most prominent of these are Feko, CST Microwave Studio, ANSYS HFSS, Zeland IE3D, and GRASP 10. The pros and cons of these applications are as follows:

- Feko: Feko has a MoM solver which has a higher accuracy than FDTD solvers, it is made accessible to students by the University of Pretoria, it has an easily accessible API that can be used to design and simulate microstrip antenna geometries automatically, it has a very intuitive interface, and it has a rather large community that utilize it which makes troubleshooting easier.
- CST: The CST solver is based on a method that is similar to FDTD, and has an easy to user interface that can be used to include very fine details in a given geometry. CST is capable of obtaining requested wide band parameters within a single execution due to the fact that its solver starts in the time domain and then converts the results to the frequency domain via Fourier Transform afterwards.
- HFSS: HFSS has an easy to user interface. The HFSS solver is based on the Finite-Difference Time-Domain (FDTD) method which means that the accuracy of the simulations is slightly less than for solvers that use MoM.
- Zeland IE3D: The Zeland solver is based on MoM. However, the user interface for Zeland isn't easy to use and is unsuited to fine details.

2.2.2 Surrogate Models

When choosing the surrogate model that would be used in the implementation of the software solution, the following surrogate models were considered:

- Artificial Neural Network (ANN)
- Support Vector Machine (SVM)
- Kriging

Artificial Neural Network

The benefits of using an ANN are as follows:

- The implementation is mathematically simple and therefore should require less time in order to reach a working solution
- There is a large scientific community that utilizes ANN's which means there should be a rather large amount of documentation concerning the subject
- The online training time for a shallow ANN is relatively low
- The prediction time for an ANN is low
- There exists several different training techniques that

With the limitations of an ANN being as follows:

- An increase in data samples as well as an increase in the dimensionality of the data samples increases the execution time
- Suffers from over-fitting
- Larger data sets provide higher precision
- Can get stuck in local minima during training

Support Vector Machine

The benefits of using an SVM are as follows:

- SVM's work well when the dimensionality of the data set is larger than the number of samples in the data set

With the limitations of the SVM being as follows:

- Online training is somewhat slow and complex
- Prediction time is somewhat slow
- The model is mathematically complex and therefore will likely require more time in order to reach a working solution
- Multiple SVM's would be required if multiple outputs need to be optimized simultaneously
- The regression problem would need to be simplified into a classification problem
- Choosing an appropriate kernel function can be difficult

Kriging

The benefits of using a Kriging model are as follows:

- The Kriging model is slightly more accurate than the ANN
- There is a large scientific community that utilizes Kriging models which means there should be a rather large amount of documentation concerning the subject

With the limitations of the Kriging model being:

- The dimensionality of the problem that the Kriging model is applied to needs to be high in order to account for the higher execution time
- The Kriging models is mathematically complex and therefore will likely require more time in order to reach a working solution

2.2.3 Data Acquisition

The method of data acquisition has a large impact on how the software solution is implemented. One method of collecting data with which to train the surrogate model would be to run several simulation over a long period of time and then to save the data from all those simulations. That data can then be used during the optimization process to train the surrogate model whilst leaving out data samples that are outside of the trust region at that point in time. This method would be a very good solution, were it not for the fact that the data required to train the surrogate model would need to be collected before the use of the software solution. This would mean that if the data had not been collected for a certain frequency before the optimization process then the optimization process would first need to be delayed in order to collect data. This would however likely reduce the number of simulations performed during the optimization process due to the fact that the data set would already exist and wouldn't need to be created at run time.

The second method of data acquisition would be to create the data set during the optimization process through the use of simulations. Whilst this would likely require a much larger set of simulations it would increase the flexibility of the solution in the sense that no preparation would be required before the optimization process. In other words, the optimization process could be performed for any frequency with the only penalty being an increase in the number of simulations and therefore the execution time.

2.3 Preferred Solution

With regards to the design alternatives the preferred solutions are as follows:

- Simulation Software: Feko has been chosen as the simulation software due to the fact that Feko has a well documented API and is readily made available by the University of Pretoria

- Surrogate Model: The ANN has been chosen as the surrogate model due to the large amount of resources available, the shorter implementation time, as well as the fact that the student has a small amount of experience in working with ANN's
- Data Acquisition: The method of data acquisition chosen is the latter choice where all the data used to train the surrogate model will be obtained at run time through the use of simulations. This choice was made due to the fact that the former choice would require a period of time to create a database before the optimization process, which would both the purpose of the time constraint for the current problem as well as make the solution less viable in a real world scenario

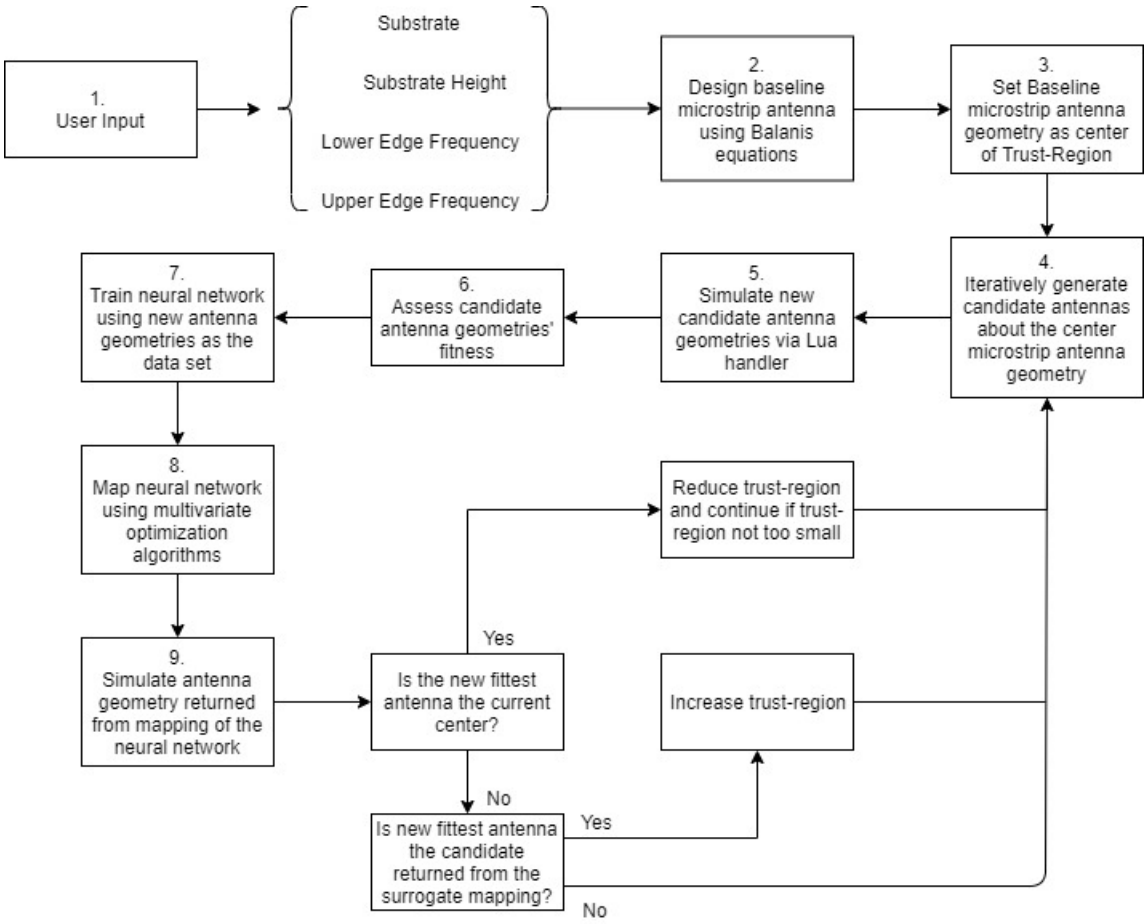


Figure 5.
System Flow Diagram

3. Design and implementation

The design and implementation of the software solution is separated into the following separate functional software units:

- The surrogate model
- The data container
- Multivariate optimization algorithms
- The optimization handler
- The surrogate handler
- The Lua interface handler
- The antenna simulation handler
- The antenna optimization handler

The above functional software units create a hierarchy where the top most functional unit uses several of the lower units but the lower units are highly unlikely to use functional units that are located higher in the hierarchy. The hierarchy of the software solution is as follows:

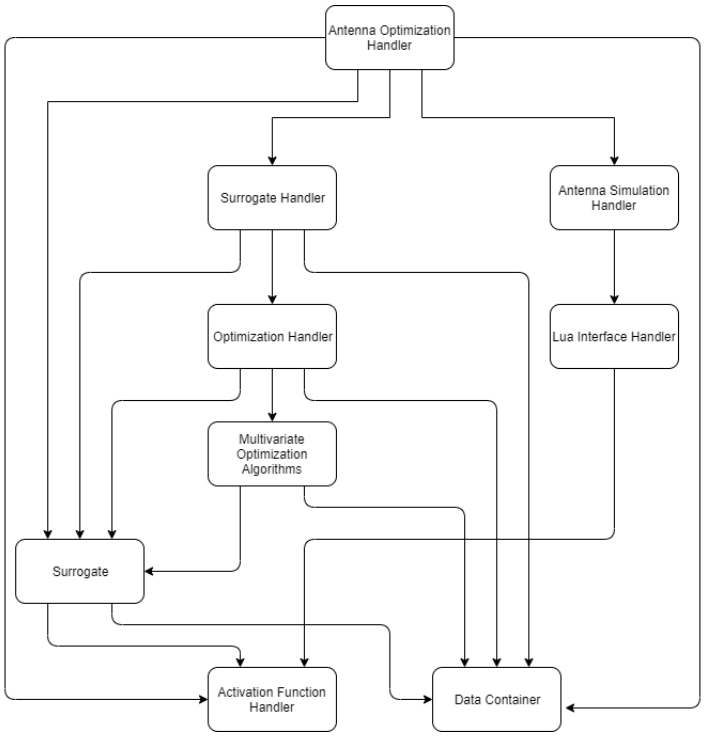


Figure 6.
Software Solution Architecture

Furthermore, let us consider the miniaturization of a microstrip patch antenna through the use of a slotted ground plane with a single rectangular slot. If we consider the four coordinates of the rectangular slots as its parameters then it's simple matter to increase the parameters of the single rectangular slot by dividing its edges repeatedly. This process of dividing the edges of the rectangular slot can be repeated an infinite amount of times until there exists an infinite number of coordinates that make up the single rectangular slot. It should then be possible to further optimize the microstrip patch antenna by optimizing each of the points on the single rectangular slot. While this would mean that slot is no longer rectangular and could therefore be more difficult to interpret it would allow for a far more flexible solution.

With this concept in mind the implantation of the slotted ground plane in this software solution is implemented in such a manner that it is possible to add additional slots to the ground plane. The addition of slots to the ground plane is then similar to the division of the edges of the single rectangular slot in the sense that a single simple slot could be expanded to a much more complicated slot which yields a far better result.

3.1 Surrogate Model

The surrogate model chosen for this project is a full-connected artificial neural network (ANN). An ANN works by taking a certain number of input and output parameters. Each set of input parameters, further referred to as a sample, is then propagated forward from the input layer of the ANN into the hidden layers of the ANN. An ANN can have multiple hidden layers of varying length. The data from each hidden layer is then propagated forward to the next hidden layer until only the output layer is left. After propagating the data from the final hidden layer to the output layer, the output of the neural network can then be easily observed. Using the output of the neural network from forward propagation, the error of the output can then be obtained by comparing the the current output of the ANN to the expected output. This error can then be used to calculate the errors of the weights of each layer in the ANN. Using the error the weights in the ANN can then be updated in order to improve the error of the ANN for that specific input-output pair. Performing multiple iterations of forward and backward propagation for the input-output pairs of an entire data set can then be seen as training the ANN.

The surrogate model functional unit will be used to predict the minimum of an objective function, where the minimum should then be an improvement on the current center microstrip antenna.

3.1.1 Geometry

The geometry of the neural network can be broken down into the following integral components:

- The input layer
- The output layer
- The weights
- The hidden layers
- The bias nodes

The Input Layer

The input layer of the ANN is very simplistic in the sense that it consists of a predefined number of nodes. Each of the nodes in the input layer will then be used to represent a unique parameter within the data set. For example, if a data set's input parameters consists of a color and a weight then the input layer for that ANN will consist of two nodes. For each sample in the data set the color input parameter will be fed into a specific node and never the other node, and similarly the weight input parameter will be fed into the unused input node and never the other.

The Output Layer

The output layer of the ANN is then very similar to the input layer. Just as with the input layer the output layer consists of a predefined number of nodes where each nodes is used to represent a single output parameter within the provided data set. That is, if a data set's output parameters consists of two classes, the output layer of the ANN will then consist of two nodes. One node will be used to represent whether or not the input parameters for a certain sample represent an apple and the other output will be used to indicate if the input parameters represent an orange.

The Weights

The weights within an ANN are the means by which data traverses from one layer to the other. This is accomplished by connecting each node in a layer to each node in the following layer where each of these connections is then called a weight. Each weight within an ANN will be assigned a numerical value during initialization then during training the weights will be iteratively adjusted in order to improve the performance of the ANN on a given data set. The method through which the data is propagated through the layers in an ANN is further explained in Section 3.1.2.

The Hidden Layers

The hidden layers within an ANN are positioned between the input layer and the output layer. Each of the multiple hidden layers within an ANN can have a varying amount of nodes and unlike the input and output layers the nodes within a hidden layer do not necessarily represent any specific parameter within a data set. As was previously mentioned each node within a hidden layer will be fully connected to both the previous layer and the next layer within the ANN via a layer of weights.

Bias Nodes

The bias nodes within an ANN are then implemented differently from the nodes within the input, output, and hidden layers. Instead of making up a layer within the ANN, the bias nodes are simply appended to the end of the input and hidden layers. That is, they add an extra node to each of these layers. However, unlike the rest of the nodes in the input and hidden layers the bias nodes are not fully connected. Instead the bias nodes are only connected to the next layer within the ANN which means that information cannot be propagated from the input parameters of the ANN to these bias nodes. Instead, only the value that they have been initialized with can be propagated forward into the following layers. The purpose of this will be further discussed in Equation 7.

Implementation

The full implementation of the geometry of the ANN can then be seen in Figure 7 below where a fully connected ANN is depicted with the following characteristics:

- Input Nodes: 2
- Output Nodes: 2
- Hidden Layers: 2
- Nodes per Hidden Layer: 4
- Bias Nodes: Active

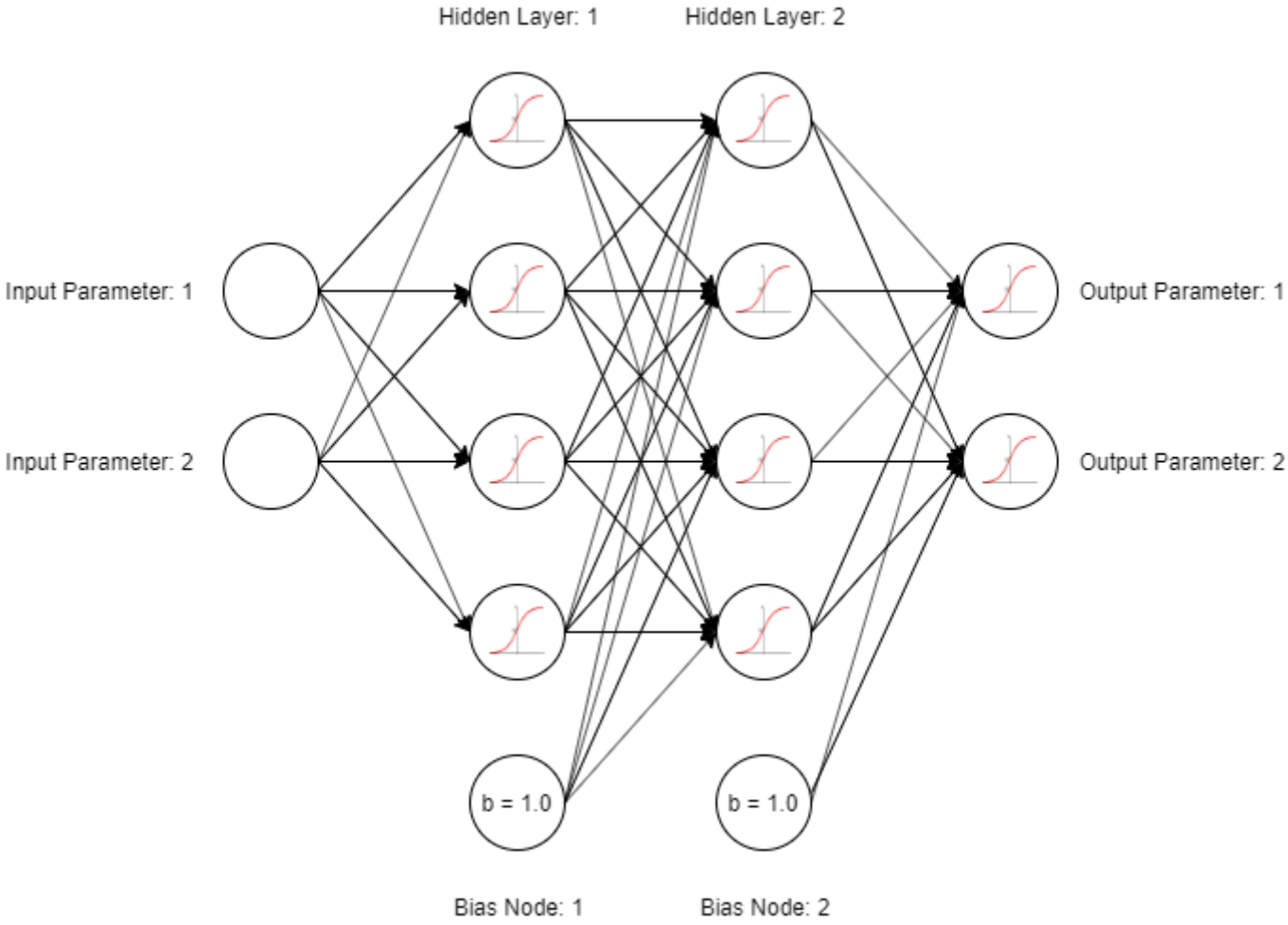


Figure 7.
Fully Connected Artificial Neural Network with Bias Nodes

In order to increase the flexibility of the ANN functional unit it was decided that the input layer, output layer, and hidden layers should all be capable of being automatically initialized during run time. This drastically improves the flexibility of the ANN functional unit by removing the constraint of requiring user input in order to change the number of nodes in the input layer.

3.1.2 Training Method

Forward Propagation

As was previously mentioned in-order to get the output of an input from a neural network the data from the input layer of the neural network must be propagated through the following layers of the ANN until the data reaches the output layer. This process is commonly known as forward propagation.

Let us consider the following arbitrary ANN geometry:

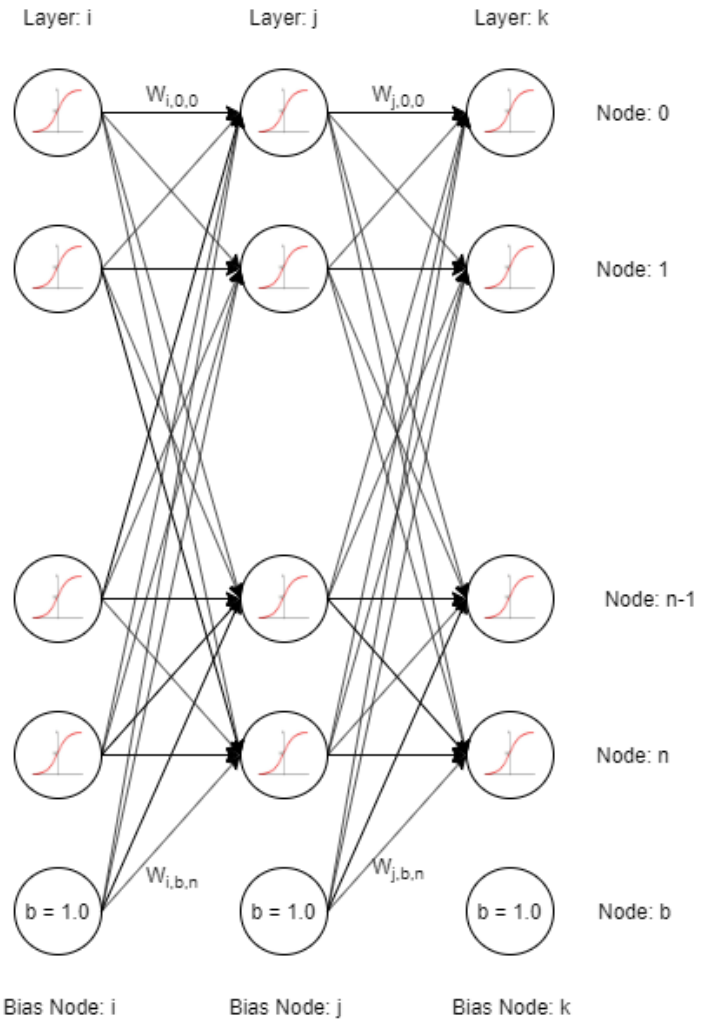


Figure 8.
Arbitrary ANN Geometry

In order to propagate the information from the I layer of the ANN to the first node in the J layer we sum the values of each node in the I layer multiplied by the weight leading from that node to the first node in layer J. That is:

$$A_{j,0} = \sum_{n=0}^{N} A_{i,n} \times W_{i,n,0} \tag{1}$$

Where:

- $A_{j,m}$ is the output of Node m in Layer j
- $W_{j,n,m}$ is the weight in Weight Layer j, leading from the Node n to Node m
- N is the number of nodes in Layer i

However, when we let the output of any given node simply be the sum of the previous nodes it becomes very difficult for the ANN to learn the data set since each node will simply be sum of the input parameters to an extent. Due to this an activation function is used on the input of each node, bar the input nodes, to get the output of that node. The purpose of this activation function is to essentially decide whether that node is on or off. This can best be described through the use of the logistic function as an activation function, where the logistic function is defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

The logistic curve is then as follows:

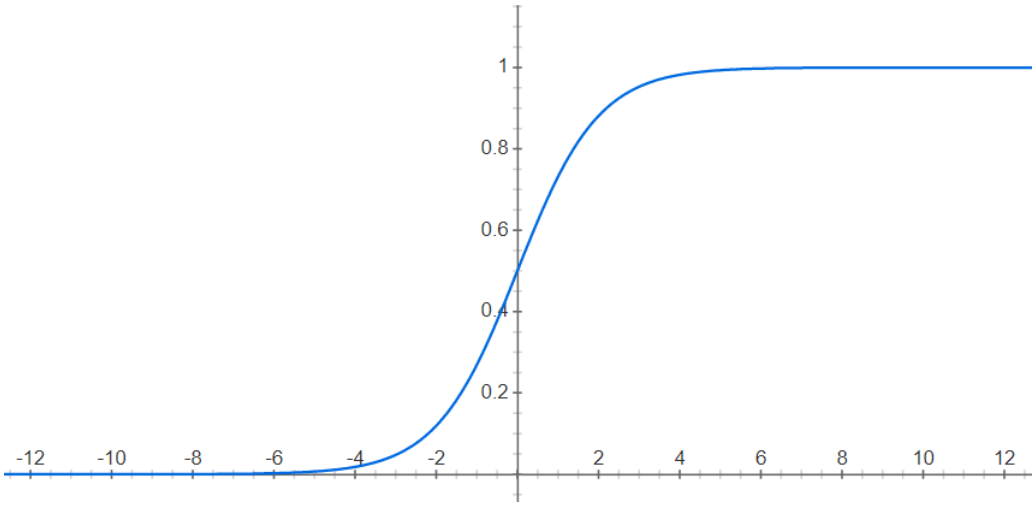


Figure 9.
Logistic Function

As can be seen in Figure 4 the output of the logistic function has the following properties:

$$\lim_{x \to -\infty} f(x) = 0 \tag{3}$$

$$\lim_{x \to \infty} f(x) = 1 \tag{4}$$

Therefore when this activation function is used as the sum of the inputs to a node become smaller then node's output tends towards zero, where as when the inputs to a node become larger the node's output tends towards one. This can be expressed as follows:

$$A_{j,0} = f\left(\sum_{n=0}^{N} A_{i,n} \times W_{i,n,0}\right) \tag{5}$$

Note now that the logistic function is centered around $x = 0.0$. This characteristic in turn means that the logistic function will be suited to classifying data that is centered around $x = 0.0$, however when the data is not centered around $x = 0.0$ the logistic function be comes a lot worse at classifying the data. The solution to this problem is to simply bias the logistic function. That is:

$$f(x-b) = \frac{1}{1+e^{-x+b}} \tag{6}$$

The effect of this biasing can be seen in the figure below for a value of $b = 5.0$:

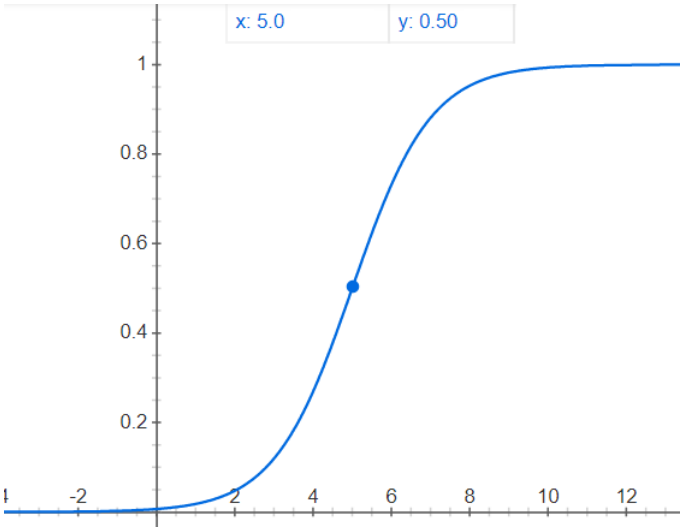


Figure 10.
Biased Logistic Function with b=5.0

This is where the functionality of the bias node plays an important role. If we set the value of each bias node to $A_{j,bias} = 1.0$ then the output of the nodes is changed to the following:

$$A_{j,0} = f\left(\left(\sum_{n=0}^{N} A_{i,n} \times W_{i,n,0}\right) + A_{i,bias} \times W_{i,bias,0}\right) \tag{7}$$

Since there are no weights leading to the bias nodes, the value of the bias nodes will always remain $A_{j,bias} = 1.0$. It should be noted at this point that due to the learning nature of the ANN, these bias nodes will very rarely bias the activation function by a value of 1.0, instead the bias node will bias the activation function by a value of $A_{j,bias} \times W_{j,bias,m}$. What this then means is that due to the learning nature of the ANN the weights connected to the bias node will be updated

during the back propagation process. This change in the weights of the bias node will then change the value by which the data for each node is biased which should center the activation function of the node to best separate the data being propagated to it.

It can be noted that the sum of the inputs to each node is saved due to the fact that it is used in the back propagation process that is discussed below.

Backward Propagation

The backward propagation utilized during the learning process of the ANN consists of calculating the error of outputs of the ANN compared to the expected outputs of a sample. The error for the outputs is commonly calculated as follows:

$$E = \frac{1}{2}\left(expected_m - A_{j,m}\right)^2 \tag{8}$$

Consider the following example:

	Node 0	Node 1	Node 2
ANN Output	-0.5	0.2	0.3
Expected Output	0.0	1.0	0.0
Error	0.125	0.72	0.045

In order for the ANN then to accurately adjust its weights due to this error it is necessary to calculate the error of each weight within the ANN geometry. The error of the weights is calculated as follows:

$$\frac{dE}{dW_{i,n,m}} = A_{i,n} \times \delta(A_{j,m}) \times \frac{dE}{dA_{j,m}} \tag{9}$$

$$\delta(A_{j,m}) = f'\left(\left(\sum_{n=0}^{N} A_{i,n} \times W_{i,n,m}\right) + A_{i,bias} \times W_{i,bias,m}\right) \tag{10}$$

From Equation 10 we can see that $\delta(x)$ is the derivative of the activation function used for any given node. As was mentioned in the preceding section, the sum of the inputs to each node is saved for precisely this scenario since there are many activation functions for which the derivative is difficult if not impossible to get using only the output of the original activation function.

Using the sum of the inputs to each node it is then easy to obtain $\delta(x)$ for any node. Two of the terms in Equation 9 are then relatively easy to obtain since the term $A_{i,n}$ is simply the current output of the relevant node. The last term $\frac{dE}{dA_{j,m}}$ is then the error of the node specified by the index j,m. For the output layer, obtaining this error is simple as it can be obtained through the use of Equation 8. However, for nodes in the hidden layer of a neural network this is slightly more complicated. The error of nodes within the hidden layer are obtained as follows:

$$\frac{dE}{dA_{j,m}} = \sum_{n=0}^{N} W_{j,m,n} \times \delta(A_{k,n}) \times \frac{dE}{dA_{k,n}} \tag{11}$$

As can be seen in Equation 11 the error of a node within the hidden layers of an ANN uses the errors of the nodes in the layers that follow it. Since we have the errors of the output layer it is simply a matter of calculating $\frac{dE}{dA_{out,n}}$ as follows:

$$\frac{dE}{dA_{out,n}} = \frac{d}{dA_{out,n}} \times \frac{1}{2}\left(expected_n - A_{out,n}\right)^2 \tag{12}$$

$$\frac{dE}{dA_{out,n}} = expected_n - A_{out,n} \tag{13}$$

Once the error of the output nodes have been calculated, the errors of the nodes in the preceding layers can easily be calculated through simple substitution of the appropriate values into Equation 11. It is then possible to iteratively calculate the error of the nodes in each preceding layer of the ANN geometry. Once the error of all the nodes, bar the input layer nodes, have been obtained it is similarly possible to calculate the error for each weight within the ANN.

Once the errors for each weight in the ANN geometry have been calculated the weights are updated as follows:

$$W_{j,m,n} = W_{j,m,n} + \mu \times W_{j,m,n} - \gamma \times \frac{dE}{dW_{j,m,n}} \tag{14}$$

Where:

- μ is the momentum. The momentum is used to aid the back propagation from falling into local minima. This is accomplished due to the fact that once a weight starts moving in a particular direction in weight space this term allows it to keep moving in that direction. The default value used for this term is $\mu = 0.001$ but can easily be changed by the user.
- γ is the learning rate. The learning rate simply scales the error being applied to the weight in order to prevent the weight from moving too rapidly in any particular direction in weight space. The default value used for this term is $\gamma = 0.05$ but can easily be changed by the user.

The back propagation process can then be implemented as follows:

```
for each layer
    for each node in current layer
        calculate delta(A)

for each layer from output to input
    for each node in current layer
        calculate E(A)
```

```
for each layer
    for each weight in current layer
        calculate E(W)
        W = W + μW - learning rate * E(W)
```

Training Process

The training process for any given data set is then as follows:

```
fitness = infinity
weights = None

iBatchSize  = count( dataSet ) / iNumberBatches

for epochs:
    for iNumberBatches
        for iBatchSize
            dataSample = dataSet.getRandomSample()
            propagateForward( dataSample.input )
            propagateBackward( dataSample.output )

        tmpFitness  = getFitness( dataSet, testingSet )

        if ( tmpFitness < fitness ):
            fitness = tmpFitness
            weights = getWeights()

        else if (tmpFitness > 5 * fitness):
            break

setWeights( weights)
```

During the training process of the ANN as seen in the pseudo-code above it can be noted that the fitness of the ANN is determined after each training batch. This fitness value is calculated as follows:

$$fitness = E_{\%}(training) \times E_{\%}(testing) + c_1 \times E_{\%}(training) + c_2 \times E_{\%}(testing) \qquad (15)$$

Where:

- $E_{\%}(x)$ determines the percent of data samples in the data set, x, that have a Root-Mean-Square Deviation (RMSD) greater than a predefined error margin

The fitness value from Equation 15 is then used to determine if the the ANN has improved. If the fitness of the ANN has improved the current weights of the ANN are saved until either the

end of the training process or until the fitness of the ANN improves again. However, if the ANN fitness has increased by a certain amount, it is deduced that the ANN is over-fitting the weights which would lead to an increase in the fitness in which case the current epoch is ended early. At the end of the training process the weights of the ANN are then reset to the last weight set that had the lowest fitness value.

In Equation 15 it can be seen that the percent of inaccurate data samples in the training and testing data set are multiplied by scalars c_1 and c_2, respectively. This is done in order to bias the training process toward one of the data sets. For instance if $c_1 = 1.0$ and $c_2 = 0.0$ then the fitness of the ANN will tend towards 0.0 as $E_{\%}(training)$ tends towards 0.0 regardless of the value of $E_{\%}(testing)$ and vice versa for the testing data set. With this biasing the ANN can easily be slightly biased to perform better on unseen data samples as opposed to the data it is being trained on. Thus through the use of these scalars in determining the fitness of the ANN the ANN is further prevented from over-fitting.

Additional Flexibility

In order to increase the flexibility of the surrogate model functional unit, several different activation functions have been implemented outside of the surrogate model. The surrogate model can then simply make a call to any of these activation functions should the need arise. Of all the activation functions that have been implemented $\delta(x) = tanH(x)$ was found to be the most successful activation function when performing regression analysis.

Additionally, the ANN functional unit was made more flexible by providing easy access to the following settings within the unit at run time:

- The activation function being used, default: tanH(x)
- The use of bias nodes, default: True
- The value that bias nodes are initialized to, default: 1.0
- The use of momentum and learning rate, default: True
- The momentum, default: 0.001
- The learning rate, default: 0.05
- The number of epochs, default: 30
- The batch size, default: 20

3.1.3 Regularization

Due to the problem of over-fitting posing such a large problem in neural networks several methods of regularization have been developed in order to further prevent over-fitting. The following regularization methods have been implemented in the ANN functional unit:

- L1 Regularization

- L2 Regularization
- Node Drop Out
- Weight Decay

L1 Regularization

L1 regularization, commonly referred to as lasso regularization, modifies Equation 14 as follows:

$$W_{j,m,n} = \begin{cases} W_{j,m,n} + \mu \times W_{j,m,n} - \gamma \times \dfrac{dE}{dW_{j,m,n}} - \gamma\lambda_1 & , W_{j,m,n} > 0 \\ W_{j,m,n} + \mu \times W_{j,m,n} - \gamma \times \dfrac{dE}{dW_{j,m,n}} + \gamma\lambda_1 & , W_{j,m,n} < 0 \end{cases} \tag{16}$$

As can be seen in Equation 16 the term $\gamma\lambda_1$ is either subtracted or added to the weight along with the error and momentum terms. Since this term does not take into consideration the actual value of the weight the effect is that during training any weights that do not play a large role in producing the output will tend towards zero. This means of regularization then essentially removes weights that play a lesser role in determining the outputs for a given input sample, and therefore performs feature selection in the data set.

L2 Regularization

L2 regularization, commonly referred to as ridge regularization, modifies Equations 14 as follows:

$$W_{j,m,n} = W_{j,m,n} + \mu \times W_{j,m,n} - \gamma \times \frac{dE}{dW_{j,m,n}} - 2 \times \gamma\lambda_2 \times W_{j,m,n} \tag{17}$$

As can be seen in Equation 17, unlike L1 regularization, L2 regularization takes the current value of the weight into consideration when adjusting it. Using L2 regularization then causes the weights to tend toward zero, however the closer any given weight is to zero the smaller the effect of the λ_2 will have on the weight.

Node Drop Out

Node drop out is unlike L1 and L2 regularization, in the sense that it does not directly adjust the values of the weights. Instead, node drop out forces a random amount of nodes in each layer, except for the output layer, to be excluded in the forward and backward propagation process. This causes the ANN not to rely on any given node more than another node due to the fact that any of the nodes could be excluded during forward and backward propagation.

When using node drop out the geometry of the ANN is changed from the normal shallow geometry to a deeper geometry. The reasoning for this is that due to the fact that there are fewer nodes to use during training the ANN needs a few extra layers to create additional paths for information propagation. Additionally, the fact that there are fewer nodes being used causes the learning rate to seem larger than it is. This is due to the fact that usually the error of a node would be spread across all of the weights leading to it; now the error is spread across fewer weights. Due to the fact that the learning rate is seemingly increased a means of preventing

the weights from becoming too large should be used. In this ANN the user can easily use L2 regularization or weight decay in tandem with node drop out, or the user can manually lower the value of the learning rate.

The following probabilities are used when determining if any given node should be dropped from the propagation iteration:

- Nodes in the Input Layer: 20%
- Nodes in the Hidden Layer: 50%

It should be noted that bias nodes are never dropped from the propagation iteration due to the fact that they only serve to bias the activation function and do not propagate any actual input data.

Weight Decay

Weight decay is a primitive form of regularization. After backward propagation has occurred all the weights in the ANN are adjusted as follows:

$$W_{j,m,n} = W_{j,m,n} \times (1.0 - \rho) \tag{18}$$

Where the parameters ρ would commonly be $\rho = 0.001$.

3.1.4 Alternative Training Methods

Whilst there are several alternative algorithms that could be used to train neural networks, none of these alternatives could be implemented due to time constraints. These alternatives include but are not limited to:

- Newton's Method
- Conjugate Gradient
- Quasi-Newton Method
- Levenberg-Marquardt Algorithm

It was, however, decided that since multiple multivariate optimization algorithms would be implemented that it would be a simple matter to apply those algorithms to the ANN training process. For example, suppose we have an ANN with 10 weights and a single output variable. Using the sum of the RMSD for the data set, it is possible to obtain what is essentially the fitness of the current weights of the ANN. That is, the lower the sum of the RMSD for any set of weights, the better the fitness of the ANN. Using this concept the training of the surrogate model simply becomes another multivariate optimization problem. The following multivariate optimization algorithms were used to test this theory:

- Trust-Region Optimization (TRO)
- Particle Swarm Optimization (PSO)

Whilst neither of the above algorithms ever achieved an extremely high accuracy by themselves; they did reduce the overall training times for the neural network when gradient decent was used after using the multivariate optimization algorithm to obtain a better starting point for the ANN.

3.1.5 Mid-End Functionality

As mentioned in Section 3.1.4 of this report some alternative training methods were used in the implementation of the ANN. Due to the fact that these alternative training methods would need to call functions that weren't designed to be called from outside of the ANN class, it was decided that the necessary functions would be given mid-end interfaces. Where:

- Front-End: Functionality easily accessible by the user
- Mid-End: Functionality not normally accessible by the user
- Back-End: Functionality not accessible by the user

This was accomplished through the use of a randomly generated password for each instance of ANN. Using this password in conjunction with the mid-end interfaces provided, the user can call back-end functions provided the password matches the current password of the ANN instance.

The following functions were provided with mid-end interfaces:

- getWeights(iPassword:int) -> list
- setWeights(iPassword:int, lWeights) -> None
- propagateForward(iPassword:int, lInput) -> None
- propgateBackward(iPassword:int, lOutput) -> None
- setActivationFunctions(iPassword:int, **kwargs) -> None

3.2 Data Container

The role of the Data Container, further referred to as the Data Handler (DH), within the software solution is as follows:

- Provide an easy to use method of accessing a data set
- Provide an easy to use method of passing a data set between different functional units

- Provide an easy to use method of normalizing and / or standardizing a data set
- Provide an easy to use method of separating a data set into multiple unique data sets

The above functionality was accomplished through the use of the front-end functions where the following general subsections of front-end functions are of importance:

- Imports and Exports
- Data normalization
- Data standardization
- Data access
- Data set manipulation

3.2.1 General Functionality

The DH functions by keeping a set number of lists, where each list contains data pertaining to the current data set. The following lists of importance are maintained within the DH:

- Unused Inputs: A list containing the input parameters of data samples that have not been used since the last reset of the DH
- Unused Outputs: A list containing the output parameters of data samples that have not been used since the last reset of the DH
- Used Inputs: A list containing the input parameters of data samples that have been used since the last reset of the DH in the order that they were used
- Used Outputs: A list containing the output parameters of data samples that have been used since the last reset of the DH in the order that they were used
- Input Data Mapping: A list containing the minimums, maximums, medians, means, standard deviation, and range of each input parameter within the data set
- Output Data Mapping: A list containing the minimums, maximums, medians, means, standard deviation, and range of each output parameter within the data set

All the functionality for the DH is implemented through the use of the above mentioned lists.

3.2.2 Imports and Exports

The DH functional unit contains functionality to both export and import any given data set that is contained within an instance of the DH class. This allows the user to save and restore the current state of data set should it be necessary for the user to do so.

The DH functional unit also contains functionality that allows the the class to import data from a .txt, .csv, or any other file that contains plain text. Through the use of this external data sets can be imported which allows the user to use a wider sample of data sets when testing any of the functional units in the software solution that require the use of this class.

Aside from importing and exporting data from files, functionality also exists to allow the user to simply set the data for the inputs and the outputs of the class.

3.2.3 Data Normalization

Due to the fact that several different models could require the use of normalization it was decided that data normalization would be implemented as close to the data source as possible. Thus data normalization was implemented in the DH functional unit.

Data normalization is the process of shifting the median of any given data set from its current median to a desired median. After the median has been shifted, the minimums and maximums are then scaled in order for the data to cover only a desired range.

For example, suppose we have a data parameter with the following characteristics:

- Maximum: 100.0
- Minimum: 53.0

The median of the data parameter above can then be calculated as follows:

$$median = min + \frac{range}{2} median = min + \frac{max - min}{2} \tag{19}$$

$$median = 53.0 + \frac{100.0 - 53.0}{2} \tag{20}$$

$$median = 53.0 + \frac{47.0}{2} \tag{21}$$

$$median = 53.0 + 23.5 \tag{22}$$

$$median = 76.5 \tag{23}$$

if we then wish for the above data parameter to be centered at 0.0 and to have a minimum and maximum of -1.0 and 1.0 respectively the following changes would be made:

$$data_{new} = \frac{data_{old} - median}{\frac{range}{2}} \tag{24}$$

$$data_{new} = \frac{data_{old} - 76.5}{23.5} \tag{25}$$

Some examples of normalized data sample parameters for this particular example would then be as follows:

	Original Data Parameter	Normalized Data Parameter
Example 1	53.0	-1.0
Example 2	100.0	1.0
Example 3	76.5	0.0
Example 4	62.34	-0.6
Example 5	89.88	0.57

Whenever the data in any given data set is standardized the minimums, maximums, medians, means, ranges, and standard deviations of all the data parameters are saved to the relevant list mentioned in Section 3.2.1.

3.2.4 Data Standardization

Data standardization is then implemented very similarly to data normalization in the sense that for any given data parameter the data parameter is first shifted by some value and then scaled to match the current range of the data to a new range. The difference between data normalization and data standardization lies in the values used to center the data parameters as well as the value used to scale the data parameters. For data standardization the new center of the data is the mean where the value used to scale the data is the standard deviation of the data. The mean and standard deviation for any given parameter can be calculated as follows:

$$\bar{x} = \frac{\sum_{i=0}^{N} x_i}{count(x)} \tag{26}$$

$$\sigma = \sqrt{\frac{\sum_{i=0}^{N} (x_i - \bar{x})^2}{count(x)}} \tag{27}$$

Where:

- x denotes a data set with N data samples
- $\bar{x}$ denotes the mean of the data set, x
- σ denotes the standard deviation of the data set, x

Whenever the data in any given data set is standardized the minimums, maximums, medians, means, ranges, and standard deviations of all the data parameters are saved to the relevant list mentioned in Section 3.2.1.

3.2.5 Data Access

Through the use of the data lists mentioned in Section 3.2.1 it becomes very simple to provide access to data within the class. Given that any index in any list refers to both the input parameters and output parameters for a single data sample, the user simply request a data sample by providing the index of the data sample within the relevant list. Alternatively, should the user wish to request a random sample this process becomes even less complex in that the user only need request a data sample then. When a random data sample is requested the DH functional unit will randomly select an index within the list of unused data samples and provide the user with a copy of that data sample. After a data sample has been passed to the user that data samples inputs and outputs will then be moved to the used lists in order to avoid repeating data samples before the entire data set has been traversed.

The DH functional unit also provides functionality to split any given data set into two subsets. By default the data set is split into two sets with the first subset containing 75% of the original data and the second subset containing the remaining 25% of the original data. This allows the user to easily create data sets used for training, testing and validating in the case of the ANN.

3.2.6 Additional Functionality

The following additional functionality has been implemented in the DH functional unit.

Gaussian Noise Injection

Gaussian noise injection is another form of data regularization. It was decided that this means of data regularization would be implemented in the DH functional unit as opposed to the being implemented in the ANN functional unit like the other forms of regularization. This was chosen due to the fact that Gaussian noise injection is a regularization technique that works directly with the data as opposed to affecting the geometry or the parameters within the ANN functional unit.

Gaussian noise injection is then implemented by generating random noise about a data sample within a given standard deviation. That is:

$$4x_i = Gauss(x_i, \sigma) \tag{28}$$

Where:

- x is a data set with N data samples
- σ is the standard deviation of the noise about the center data sample, x_i

Gaussian noise injection works by preventing the ANN from learning the exact input-output mappings which is extremely valuable in a small data set since this tends to occur due to the nature of the problem. Thus by using Gaussian noise injection it becomes possible to essentially fool the ANN functional unit into thinking the data set is larger than it actually is which prevents over-fitting.

The default value for the standard deviation being used for noise injection is $\sigma = 0.02$.

3.3 Multivariate Optimization Algorithms

The multivariate optimization algorithms are commonly grouped into swarm based algorithms, evolutionary algorithms, and numerical algorithms. Swarm based optimizers are optimizers that generally act according to principles such co-operation or competition. That is, the particles within a swarm usually either aid each other in the search for a better minimum through concepts such as information sharing or compete against each other in order to get reach a better minimum. Swarm based optimizers also typically contain a set amount of particles that move every iteration and aren't typically killed off to create new particles that could possibly be better candidates as is commonly the case with Evolutionary Algorithms. The following multivariate optimization algorithms have been implemented in order to add surrogate mapping functionality:

3.3.1 Particle Swarm Optimization

The PSO algorithm was implemented as is described in [2]

3.3.2 Trust-Region Optimization

The TRO algorithm was implemented as is described in [1]

Of these two algorithms it can be noted that PSO is a swarm-based algorithm whilst TRO is a numerical algorithm.

3.3.3 Optimization Handler

The purpose of the Optimization Handler (OH) within the software solution is to accept a multivariate problem and then to use the available optimization algorithms to map the objective function of the passed multivariate problem. The following multivariate problems can be passed to the OH:

- Mapping of a surrogate, where the surrogate's output is the objective function to be mapped
- Mapping of the weights of an ANN, where the fitness of the surrogate is the objective function to be mapped as mentioned in Section 3.1.4

As was mentioned in Section 3.3 the following multivariate optimization algorithms are available for the OH to use when mapping the above mentioned objective functions:

- PSO
- TRO

With that in mind it should be clear that this functional unit's purpose was simply to act as the middle-man during the use of the implemented multivariate optimization algorithms. However, when the functional unit was initially implemented it was noticed that Python was only using a single core of the Central Processing Unit (CPU) during the execution of any of the optimization algorithms. In an effort to increase the speed of the optimization algorithms it was proposed that multi-threading be used in order to run multiple threads of optimization algorithms simultaneously. Whilst this would technically not increase the speed of any one given optimization algorithm it would technically allow the OH to execute as many optimization algorithms as the computer running the software has CPU cores. That is, for a 4-core CPU computer multi-threading should allow for 4 simultaneous executions of separate and unique optimization algorithms.

Unfortunately, however, this was not the case. In reality the multi-threading library used in python did not create separate threads that could use the free cores of the CPU. Instead, the library creates separate and unique threads that could all run their own optimization algorithms but all of these threads still belong to the same process. This fact at the end of the day meant that all the new threads were still running on the same CPU and with four threads running simultaneously the optimization algorithms were therefore simply four times slower.

Eventually the solution to the problem was found in using multi-processing as opposed to multi-threading. Through the use of multi-processing the OH is capable of creating separate and unique processes each capable of executing their own optimization algorithms. Each of the child processes then returns the results of the optimization algorithm to the parent process which in turn evaluates the results and picks the candidate with the lowest fitness from the several different optimization algorithm mapping executions.

The full implementation for the OH can be found in the Technical Document. The OH can then be understood through the use of the following pseudo-code:

```
Setup Global Variables
Setup Local Variables

while (True):
    for i in range(4):
        if (thread[i] == None):
            Create a new event
            Create new thread dictionary
            Create a new queue
            Create new thread
            Set thread variable
            Set training flag and increment thread id

        else:
            if (thread[i].is_set() == True):
                Clear event
                Append data
                if (iThread_ID < iThread):
                    Create new thread dictionary
                    Create a new thread
```

```
                    Set thread var
                    Increment thread id

    if ( len( results ) == 16):
        Exit loop

for i in range(0, len( results )):
    if (thread[i].fitness < fFittest):
        Set new best candidate

return best candidate
```

3.4 Surrogate Handler

The Surrogate Handler (SH) functional unit has very similar functionality to that of the OH in the sense that it simply acts as the middle-man for training and mapping an objective function. Specifically the SH takes an instance of DH as an input and then trains multiple separate surrogate models using the passed DH. Each of the surrogate models trained then has their overall fitness evaluated by taking the sum of the RMSD across all of the data samples in the passed data set as well as it's accuracy across all the data samples in the passed data set. The overall fitness of a surrogate is then calculated as follows:

$$fit_{RMSD} = \sum_{i=0}^{N} RMSD(x_i) \tag{29}$$

$$fit_{Acc} = \frac{\sum_{i=0}^{N} accurate(x_i)}{count(x)} \tag{30}$$

$$fit_{tot} = fit_{RMSD} * (1.1 - fit_{Acc}) \tag{31}$$

Where:

- x is a data set with N items
- $$accurate(x_i) = \begin{cases} 1 & , \quad RMSD(x_i) <= \rho_d \\ 0 & , \quad RMSD(x_i) > \rho_d \end{cases} \tag{32}$$
- ρ_d is the error margin that the RMSD of any data sample, x_i must fall within in order for the surrogate model output to be considered correct

The surrogate training is carried out by passing the provided data set in the form of the DH to the ANN. The ANN then trains itself using either normal gradient decent or through the use the multivariate optimization algorithms as was outlined in Section 3.1.4.

After the training of a pre-defined number of surrogates has taken place the surrogate handler will outsource the mapping of the surrogates with the highest total fitness as determined in Equation 31 to the OH outlined in Section 3.3.3. After the fittest surrogates have been mapped their results are stored in a public variable that is easily accessible to either the user or the functional unit that create the SH. Through the implementation of this flexibility the results of both the surrogate training process and the surrogate mapping process are always readily available.

3.5 Lua Interface Handler

The Lua Interface Handler (LIH) plays an extremely integral part in the software solution. That is, the LIH is the interface between the software solution and solver, namely Feko, being used to simulate the randomly generated antenna geometries. The LIH was designed to be as flexible as possible whilst providing access to as much of the functionality of Feko as possible.

The LIH provides access to the user through the use of the Feko API [9]. However, this is further complicated by the fact that the Feko API doesn't use any of the more common scripting languages such as:

- Matlab
- Python
- Java
- C#

Instead, the Feko API utilizes an in-house scripting house that was developed by Feko, called Lua. This scripting language closely resembles Python, but not enough so for the scripts simply to be written in Python. With this in mind the LIH was designed not simply to interact with the Feko API, instead the LIH creates and stores a list of commands in the Lua scripting language. When the user terminates the script the LIH then creates a Lua script using the stored commands and then proceeds to run that script through the execution of a Feko via a terminal command. It should then be clear that the flexibility of the antenna geometry is directly related to the ability to simulate antennas through the use of the LIH functional unit. With this in mind and without going into too much detail the following functionality has been implemented as the front-end for the LIH functional unit:

1. exportLua(**kwargs) : This function exports the entirety of the Lua code stored locally within the current instance of the LIH functional unit. This function also serves to execute the exported Lua script via a terminal command to the Feko API.
2. setUnits(**kwargs) : This function is used to set the units being used in the creation of the antenna geometry in the Feko simulation as a difference between cm's and mm's would obviously make a very big difference in the outcome of a simulation
3. newAntenna(**kwargs) : This function creates a new antenna project in the current LIH instance. This is of utmost importance due to the fact that the Feko application takes nearly 20s to launch which is approximately the same time as running a single simulation.

Therefore if we can run multiple simulations whilst only having to launch the application once a large amount of times is saved

4. deleteAntenna(**kwargs) : This function deletes the current antenna project in the current LIH instance

5. saveAntenna(**kwargs) : This function sets the directory in which the current antenna project should be created and thus simulated. This is more important than it might seem at first glance due to the fact that if for instance 45 simulation are run in the same directory then it could become very difficult to keep track of which antenna project is which. Therefore organizing each project into it's own directory is considered reasonably important

6. simulateAntenna(**kwargs) : This function sets the simulation parameters for the current antenna project as well as if at the end of the construction of the antenna geometry in Feko the model should be simulated

7. closeAntenna(**kwargs) : This function closes the current antenna project, this is required due to the fact that multiple antenna projects are created in a single LIH instance

8. setSimulationFrequency(**kwargs) : This function sets the frequency parameters for the simulation of the current antenna project

9. setSimulationMesh(**kwargs) : This function sets the mesh parameters for the simulation of the current antenna project

10. createNewMedium(**kwargs) : This function creates a new medium as specified by the parameters in **kwargs. This is primarily used for the creation of substrates at the moment

11. setFacemedium(**kwargs) : This function sets the medium of the specified face. This is primarily used after the use of the Union modification to reset the radiating plane and ground plane to that of a conductor since the union modification changes all faces' medium to the medium used as the substrate

12. setSolidMedium(**kwargs) : This function sets the medium of a solid and is primarily used to set the medium of the substrate

13. createNewSolid(**kwargs) : This function creates a new solid and is primarily used to create the substrate

14. createNewSurface(**kwargs) : This function creates a new surface and is used to create the ground plane, radiating plane, slots, and the feed strip. This function allows for the creation of rectangular, triangular, elliptical, and polygonal surfaces provided all the correct parameters are provided via **kwargs

15. createNewModification(**kwargs) : This function creates a new modification to the antenna model and is primarily used to perform a union on the antenna model before simulation

16. createNewPort(**kwargs) : This function can be used to create a new edge port or a new line port at any edge or line on the antenna model, respectively

17. createNewSource(**kwargs) : This function creates a new voltage source at the specified port with the specified amplitude and phase
18. createNewRequest(**kwargs) : This function is used to create a new simulation request and is used primarily to request the radiation patterns of the E and the H plane of the antenna model

3.6 Antenna Simulation Handler

The Antenna Simulation Handler (ASH) takes a list of antenna geometries as an input and then through the use of the LIH functional unit simulates the aforementioned antenna geometries. Once the antenna geometries have been simulated, the ASH functional unit can be used to extract the simulated outputs of the antenna geometries from the respective files. Since the majority of the simulation process is done through the LIH functional unit this work won't be repeated in this Section. The output parameters of any given antenna geometry are then extracted as follows:

```
for count( candidates ):
    lData = readLines( candidate )

    while (i < len( lData) ):
        if ( lData[i] = gain):
            fTmp_Directivity    = -np.inf
            fTmp_Eff            = None

            while (True):
                Increment i
                if ( lData[i] = end of gain ):
                    Increment i
                    Get efficiency
                    Exit loop

                else:
                    fTmp_Dir    = Strip new line

                    if (fTmp_Dir > fTmp_Directivity):
                        fTmp_Directivity = fTmp_Dir

            Calculate gain
            Save data to temp dictionary
            Append to actual data list
            Clear tmp variables

        else if ( lData[i] = frequency):
            lData_F = lData[i].strip( noise )
            lData_F = self.__cleanString__( lData_F )
```

```
        else if ( lData[i] = s11):
            lData_S = lData[i].strip( noise )
            lData_S = self.__cleanString__( lData_S )

        Increment i

    for i in range(0, len( lData_Actual ) ):
        if (lData_Actual[i].gain != None):
            fTmp_Gain       = lData_Actual[i].gain
            fTmp_Gain       = vActivations.logistic( 1.0 - 2.0 * fTmp_Gai
            lData_Actual[i].fitness     += fTmp_Gain

        if (lData_Actual[i].frequency < fDesired_Lower):
            fLower_Sum      += lData_Actual[i].fitness
            fLower_Samples  += 1

        else if (lData_Actual[i].frequency < fDesired_Upper):
            fDesired_Sum        += lData_Actual[i].fitness
            fDesired_Samples    += 1

        else:
            fUpper_Sum      += lData_Actual[i].fitness
            fUpper_Samples  += 1

        if (lData_Actual[i].fitness < fResonant_Fitness):
            fResonant_Frequency = lData_Actual[i].frequency
            fResonant_Fitness   = lData_Actual[i].fitness

    if (fLower_Samples > 0):
        fLower_Sum      = fLower_Sum / float( fLower_Samples )

    else:
        fLower_Sum      = 0.0

    if (fDesired_Samples > 0):
        fDesired_Sum        = fDesired_Sum / float( fDesired_Samples )

    else:
        fDesired_Sum        = 0.0

    if (fUpper_Samples > 0):
        fUpper_Sum      = fUpper_Sum / float( fUpper_Samples )

    else:
        fUpper_Sum      = 0.0

return data
```

In Step 2 in Figure **??** the .out file for the specified antenna geometry is accessed. During the simulation of the antenna geometry Feko essentially dumps all the data for the simulation in this .out file. Once the file has been accessed each line in the file is evaluated in an iterative loop until a new frequency is found. When a new frequency is found the file output continues to be evaluated iteratively, however now specific data is being searched for. The data that appears first is the data pertaining to the radiation pattern of the antenna geometry. When iterating through the radiation pattern data each line of data is first stripped of the string subsets that don't contain any useful information. Once this is completed the directivity of the antenna geometry in the direction specified in that line is evaluated. If the new directivity is the largest directivity seen thus far it is saved until the end of the radiation pattern data has been reached. Once the end of the radiation pattern data has been reached we can obtain the efficiency of the antenna geometry at the current frequency. A good approximation as to the gain can then be made using the product of the directivity and the efficiency of the antenna geometry as follows:

$$fit_G = Directivity \quad \times \quad Efficiency \tag{33}$$

The next step is then to search for the $|s_{11}|$ parameter for this frequency. After the $|s_{11}|$ parameter has been found and the string containing the parameter has been stripped of the subsets of strings that don't contain useful data we save fit_G, $|s_{11}|$, and the current frequency to a dictionary in the following format and then append the data to a list to be further processed after all the output parameters have been extracted from the .out file:

```
dFrequencyData = {
    "frequency" :   fFrequency,
    "gain":         fFit_G,
    "s11":          fFit_S11
}
```

After the extraction of all the parameters from the .out file the fitness of the antenna geometry is determined at each frequency that was simulated. This first step in determining the fitness of the antenna at any given frequency is to determine the fitness of the $|s_{11}|$ parameter at that frequency as follows:

$$fitness_{|s,11|} = logistic\left(\frac{|s_{11}| + 8}{1.5}\right) \tag{34}$$

The reasoning behind using the logistic function to determine the fitness of the $|s_{11}|$ parameter is that for any given frequency we're attempting to optimize the antenna geometry in such a manner that the $|s_{11}|$ parameter at that frequency is less than $-10dB$. That is:

$$F_{s,11} < -10dB \tag{35}$$

Since we're simply looking for a value less than $-10dB$ we can assume that a value of $|s_{11}| = -20dB$ will be just as good as a value $|s_{11}| = -100dB$. With this assumption in mind we then bias

the value of $|s_{11}|$ to be centered at $|s_{11}| = -8dB$. If we look at the graph of the logistic function in Figure 9 it is easy to note that the change in the logistic curve becomes nearly negligible at values of $x = -4.0$ and $x = 4.0$ with the most change in the logistic curve occurring at its center, $x = 0.0$. By biasing the $|s_{11}|$ parameter in Equation 34 by the specified values the change in the $|s_{11}|$ value at the values $|s_{11}| > -2dB$ and $|s_{11}| < -14dB$ are very near to negligible when considering the fitness of the $|s_{11}|$ parameter of any given frequency. We have then sufficiently prevented extremely low minima in the $|s_{11}|$ parameter from completely biasing the fitness of the antenna geometry.

The next step is then to determine the fitness of the gain at the current frequency. This is accomplished in a similar manner to that of the $|s_{11}|$ parameter, and is calculated as follows:

$$fitness_G = logistic(1.0 - 2.0 \times G) \tag{36}$$

Where:

- G is the rough approximation of the gain of the antenna geometry calculated at any given frequency, found in Equation 33

Since the gain value that the antenna should reach is 1.25 dBi it was decided that the gain would be centred at a value of $G = -1.0$ with the desired value of $G = 1.25dBi$ then having a fitness of:

$$fitness_{G=1.25} = logistic(1.0 - 2.0 \times 1.25) \tag{37}$$

$$fitness_{G=1.25} = logistic(-1.5) \tag{38}$$

$$fitness_{G=1.25} = 0.182 \tag{39}$$

Through the use of the Equation 36 we have thus biased the fitness of the approximation of the gain at the specified frequency.

The last step in determining the fitness of an antenna geometry at any given frequency is then to sum fitness of the $|s_{11}|$ parameter and the fitness of the approximation of the gain. That is:

$$fitness_F = fitness_{s,11} \quad + \quad fitness_G \tag{40}$$

This can be done quite safely since both the parameters being summed have been biased to prevent extremely large and extremely small values from creating large differences in the respective fitness values. The two values are therefore summed due to the fact that in order for the fitness of an antenna geometry at any given frequency to be considered good, both the gain and $|s_{11}|$ parameter must fall within ranges that are considered good. Only having the $|s_{11}|$ parameter fall within a good range with the gain being minuscule would be useless since the antenna clearly wouldn't radiate in that case where as having a good gain but a high $|s_{11}|$ parameter would indicate that the antenna simply wouldn't function at the desired frequency regardless of the gain.

3.7 Antenna Optimization Handler

The Antenna Optimization Handler (AOH) directly interacts with the ASH, SH, and DH covered in Sections 3.2, 3.4, and 3.6 respectively. Through the use of these functional units the AOH implements a TRO optimization algorithm that automatically miniaturizes a microstrip patch antenna geometry to operate at a user defined frequency. Since the size of a standard rectangular patch antenna is directly related to the frequency it operates at it is clear that as we miniaturize the baseline microstrip antenna the operating frequency of the miniaturized patch antenna will shift to a higher frequency than the frequency the antenna is designed for. In order to miniaturize the microstrip antenna whilst maintaining the same operating frequency a method of shifting the operating frequency is required. This is accomplished through the addition slots in the ground plane of the microstrip patch antenna. That is, ideally as as the microstrip patch antenna is miniaturized the frequency shift of the miniaturization effect can be cancelled through the use of slots on the ground plane.

3.7.1 Optimization Phases

Looking at the general method used during TRO in Section 3.3.2 we can see that in the first iteration of the algorithm candidates generated about the center antenna geometry will be generated around the baseline microstrip antenna geometry. This presents a unique problem in the sense that the baseline microstrip antenna geometry does not have any slots in the ground plane and therefore adjusting the parameters of slots within the bounds of the trust region becomes impossible. With this in mind the following solution was proposed: Instead of attempting to optimize the baseline microstrip patch antenna geometry from the start of the optimization process the it was required that the process be split into two distinct optimization phases each with their own objectives. These objectives of the optimization phases are as follows:

- Primary Phase: Create an antenna geometry with slots in the ground and radiating plane that can be used during the Secondary Phase as the center antenna geometry
- Secondary Phase: Use standard TRO to optimize the center antenna geometry generated during the Primary Phase

Primary Optimization Phase

The primary phase of the TRO process is very similar to the standard TRO algorithm. The differences between the primary optimization phase and standard TRO occur during candidate generation and after the simulation of the candidate antenna geometries.

The candidate antenna geometries generated about the center antenna geometry during this process follow the same general principle of TRO in that all the parameters of the candidate antenna geometries are adjusted about the center geometry whilst being kept within the trust region. However, after the parameters for the candidate antenna geometries have been adjusted, each candidate antenna geometry goes through a process where they either have a slot added or removed from their ground plane. It should be noted that this process is also then repeated

for the radiating plane. Through the use of this functionality we can then generate antenna geometries with slots in the ground plane which can easily be adjusted during the secondary phase of the optimization process.

After the new antenna candidate geometries have then been simulated we can determine their fitness according to the method described in 3.7.3. However, after the fitness of all the new candidate antenna geometries have been simulated we cannot use a surrogate model to map the objective function of the current trust region. The reason for this is due to the fact that during the slot addition and removal process at the end of the candidate generation process we both create and remove different parameters in the candidate antenna geometries. This means that at the end of the candidate creation process each candidate antenna geometry could possibly consist of a completely different set of parameters from the rest of the candidates. Therefore, surrogate modelling becomes impossible due to the fact that it requires that data sample parameters be grouped into a single input where any given candidate antenna geometry might not contain many of the parameters that are present in other candidate antenna geometries. With this in mind the mapping of the objective function is skipped and the new candidate antenna geometries' fitness are directly evaluated as follows:

- If any of the candidate antenna geometries' fitness is better than that of the current center antenna geometry, then increase the trust region and set the center antenna geometry to the fittest candidate antenna geometry
- If none of the candidate antenna geometries' fitness is better than that of the current center antenna geometry, then decrease the trust region and maintain the current center antenna geometry

Secondary Optimization Phase

The secondary optimization phase works functionally the same as the TRO algorithm described in Section 3.3.2, however there are some differences in the implementation of this phase. These differences in the implementation of the secondary optimization phase were implemented in order to increase the accuracy of the surrogate model. The following additional changes were added to the standard TRO algorithm:

- A subset of parameters are chosen to optimize during each iteration of the TRO algorithm
- The data from multiple iterations is saved until a new subset of parameters is chosen for optimization
- Data samples that are outside of the trust region are removed from the data set before surrogate modelling

These changes in the implementation of the standard TRO algorithm were implemented in order to reduce the dimensionality of the surrogate model and thereby increase the accuracy of the surrogate model. The adjusted TRO algorithm is then as follows:

```
for iterations / 3 :
    inputData   = []
```

```
outputData  = []

parameters  = getParameters(center)

for 3:
    candidates  = getCandidates(center, parameters)
    simulation  = simuCandidates(candidates)
    fitness     = getFitness(simulation, candidates)

    data        = decomposeData(candidates, fitness, parameters)

    inputData.extend(data.inputs)
    outputData.extend(data.outptus)

    srgCandidate    = doSurrogateModelling(inputData, outputData)
    srgSimulate     = simulateCandidates(candidates)
    srgFitness      = getFitness(srgSimulate, srgCandidate)

    if (srgFitness not lowest fitness):
        decrease trust region

        if (trust region <= 0):
            break

    else:
        increase trust region

if (trust region <= 0):
    exit function
```

The functions getCandidates(), getFitness(), decomposeData(), and doSurrogateModelling() are then further described in the sections that follow.

3.7.2 Antenna Candidate Creation

During the process of candidate generation each parameter in the antenna is randomized about the center of that parameter whilst keeping the parameter within the trust region. This was accomplished by assigning each unique parameter that any given antenna geometry could have a randomization region and a randomization range. Where:

- Randomization Region: A value in the range [0, 1] that is used to determine if the randomization of the parameter should be biased to become smaller, larger, or if it should be unbiased
- Randomization Range: The range within which that parameter is allowed to change

For the sake of making the following equations more interpretable let:

- r_1 = Randomization Region
- r_2 = Randomization Range

The randomization of any given parameter is then calculated as follows:

$$x_{offset} = x_{center} + 2r_2(r_1 - 0.5) \tag{41}$$

$$x_{new} = Gaussian(x_{offset}, r_2 \times region) \tag{42}$$

$$\tag{43}$$

As can be seen in Equation 42 the values r_1 and r_2 are used to calculate and offset which will be used as the center for a random Gaussian generator in Equation 43. In Equation 43 the current value of the trust region is then used to modify the standard deviation of the random Gaussian generator. What this implies is that as the trust region becomes smaller so does the standard deviation and by doing so the new random value generated for the parameter moves closer to that of the parameter in the center antenna geometry.

It should be noted that during the secondary optimization phase this process is only performed on the subset of antenna geometry parameters that are being optimized at any given time. By doing so those parameters are the only parameters that change and therefore they are the only parameters that affect the antenna simulations.

Slot Creation and Removal

During the slot creation and removal process, the first step is the removal of a slot from the candidate antenna geometries. This is accomplished as follows:

```
for each candidate:
    random = Uniform(0.0, 1.0)

    if (random < removal probability):
        index = RandInt(0, number of slots)

        remove(index)
```

As can be seen there exists a probability to remove a slot which means that some candidates will have a slot removed during this process where as other candidates might not have a slot removed. The slot being removed is also randomized.

The next step is then to create a new slot in the antenna candidate geometries. Before explaining this process it should be noted that any of the different types of slots mentioned in Section 14 can be created through the use of the createNewSurface function which is Function 14 on the list. This is accomplished as follows:

```
for each candidate:
    random = Uniform(0.0, 1.0)

    if (random < rectangular probability):
        create new rectangular slot

    elif (random < triangular probability):
        create new triangular slot

    elif (random < elliptical probability):
        create new elliptical slot

    elif (random < polygonal probability):
        create new polygonal slot
```

By assigning appropriate probabilities to the creation of a certain type of slot it then not only becomes possible to bias the type of slot created but also to limit the probability of a slot being created. The method of creating a slot is then very similar to the method used to randomize antenna parameters in Equations 42 and 42. For the creation of any given slot a random x-coordinate and a random y-coordinate are obtained that lie on the surface of the plane to which the slot will be added. These two coordinates are then used as the centers for determining the x and y coordinates of any slot.

3.7.3 Antenna Candidate Fitness Evaluation

Area Fitness Evaluation

The fitness of the surface are of the candidate antenna is evaluated as follows:

$$A_p = \frac{A_{candidate}}{A_{baseline}} \tag{44}$$

$$A_{fit} = Logistic\left((A_p - 0.775) * 17.5\right) \tag{45}$$

Once again through the use of the logistic function we can easily bias the fitness of the surface area of any given antenna candidate. By biasing the parameter A_p with the values provided we achieve the following fitness values:

- $A_p = 1.0 \quad \longrightarrow A_{fit} = 0.981$
- $A_p = 0.55 \quad \longrightarrow A_{fit} = 0.019$

Frequency Fitness Evaluation

The majority of the work required to obtain the fitness of the frequency spectrum was already performed in Section 3.6. At this point the only thing left to do is to bias the values of the

simulated frequencies inside the desired bandwidth compared to the simulated frequencies outside of the simulated bandwidth. That is:

$$F_{fit} = 0.2 \times F_{f,lower} + 0.7 \times F_{f,desired} + 0.1 \times F_{f,upper} \tag{46}$$

Where:

- $F_{f,lower}$: The sum of the fitness of the frequencies that are lower than the desired bandwidth
- $F_{f,desired}$: The sum of the fitness of the frequencies that are within the desired bandwidth
- $F_{f,upper}$: The sum of the fitness of the frequencies that are above the desired bandwidth

Final Fitness Evaluation

The final fitness of any given antenna geometry is calculated as follows:

$$Fitness = (c_1 \times F_{fit} \times A_{fit}) + (c_2 \times F_{fit}) + (c_3 \times A_{fit}) \tag{47}$$

Where:

- c_1, c_2, c_3 are scalars defined during the initialization of the AOH that allow us to alter the way the final fitness of an antenna geometry is calculated.

It is important at this point in time to note that the final fitness of an antenna geometry as calculated in Equation 47 makes use of both the sum and the product of the A_{fit} and F_{fit} terms. The reasoning for this is as follows:

- When using the product of the terms A_{fit} and F_{fit} the fitness could become zero if the size of the antenna is reduced to zero. This would however be highly unlikely to produce an antenna that is capable of operating within the desired bandwidth but does have the upside of being very sensitive to either of the terms being minimized.
- When using the sum of the terms A_{fit} and F_{fit} the fitness could very likely not accurately represent the fitness of the antenna geometry. For further explanation on this see the following example.

Suppose we have to candidate antenna geometries with the following fitness parameters:

Candidate	A_{fit}	F_{fit}
1	0.5	0.5
2	0.0	1.0

Looking at Table 3.7.3 it is easy to see that candidate antenna 1 is the better candidate. Whilst the second candidate may have a good area fitness the fact that it's frequency fitness is as high as it is indicates clearly that it does not operate within the desired bandwidth. If we then at the sum, product, and the implemented fitness parameters the table changes as follows:

Where:

Candidate	A_{fit}	F_{fit}	Sum	Product	Final
1	0.5	0.5	1.0	0.25	0.875
2	0.0	1.0	1.0	0.0	1.0

- $c_1 = 1.0$
- $c_2 = 0.25$
- $c_3 = 1.0$

Therefore using Equation 47 with the appropriately chosen scalars allows us to accurately determine the fitness of the candidate antenna geometries.

3.7.4 Data Decomposition

Data decomposition is simply the act of iterating through all the candidates and grouping the required parameters into arrays, thereby creating a data set from the candidate antenna geometries and their respective outputs.

3.7.5 Surrogate Modelling

The surrogate modelling function only really serves to remove data samples in the data set that lie outside of the trust region. After this has been completed the rest of the surrogate modelling process is completed by the SH which was covered in 3.4

3.8 Design summary

In Table 1 in his section the project design tasks are summarized as well as their respective implementations

Task	**Implementation**	**Task completion and report section**
Design of a baseline microstrip antenna that operates in the 900 MHz to 940 MHz frequency band.	The baseline microstrip antenna that operates within the 900 MHz to 940 MHz frequency band was initially designed by hand through the use of the equations provided in Balanis [7].	The equations used to design the baseline microstrip antenna geometry are situated in the ASH functional unit and can currently be found in the getPatch$_{D}efault(**kwargs)functionintheTechnicalDocumenta$
Selection and implementation of a surrogate model that is both fast and reliable..	The three most commonly used surrogate models were evaluated and the ANN was chosen as the surrogate model to be implemented for the software solution. The ANN was then implemented from first principles whilst keeping the flexibility of the functional unit in mind.	The three most commonly used surrogate models were evaluated in Section 2.2.2. The ANN was then implemented in Section 3.1
Design and implementation of a suitable cost function.	The fitness function for the candidate antenna geometries was implemented from first principles.	The fitness function is covered in Section 3.7.3, Equation 47
Selection and implementation of an appropriate optimization algorithm.	Two algorithms were selected and implemented as multivariate optimization algorithms, namely: TRO and PSO. These two algorithms were implemented from first principles.	The implementation of the TRO and PSO algorithms can be found in Section 3.3.2 and Section 3.3.1 respectively.
Design and implementation of the trust region algorithm.	The trust-region algorithm was designed according to the method used in [1] where alterations were made in order to increase the accuracy of the surrogate model. The TRO antenna optimization algorithm was implemented from first principles.	The implementation of the TRO antenna optimization can be found in Section 3.7

Table 1.
Design summary

4. Results

4.1 Summary of results achieved

Intended outcome	Actual outcome	Location in report
System requirements and specifications		
A miniaturized microstrip antenna that operates in the 900 MHz to 940 MHz frequency range.	A miniaturized microstrip antenna that operates in the 978.8 MHz to 1015.7 MHz frequency range.	Section 4.2.1
A miniaturized microstrip antenna with a gain of 1.25 dBi	A miniaturized microstrip antenna with a gain of 0.0 dBi	Section 4.2.2
A miniaturized microstrip antenna that has had its surface area reduced by 45% compared to the baseline microstrip antenna	A miniaturized microstrip antenna that has had its surface are reduced by 31.93% compared to the baseline microstrip antenna	Section 4.2.3
A TRO algorithm that is capable of automatically optimizing an input microstrip antenna geometry within 3 hours	A TRO algorithm that is capable of automatically optimizing an input microstrip antenna geometry within 4.7 hours	Section 4.2.4
A TRO algorithm that uses no more than 45 high-fidelity EM simulations in order to optimize a given microstrip antenna geometry	A TRO algorithm that uses approximately the equivalent of 50 high-fidelity EM simulations in order to optimize a given microstrip antenna geometry	Section 4.2.5
Field conditions		
The antenna should be within 40 meters of the transmitter in an indoor environment	The printed miniaturized microstrip antenna was tested using equipment that was within 40 meters of the antenna and in an indoor environment	Section 4.2.1 and Section 4.2.2

Table 2.
Summary of results achieved

Intended outcome	Actual outcome	Location in report
Deliverables		
The deliverable is the miniaturized microstrip antenna and the aspects to be designed are the frequency band that was chosen for the microstrip antenna to radiate over	The miniaturized microstrip antenna was printed and tested.	Section 4.2.1
The deliverable is the miniaturized microstrip antenna The miniaturization will not be directly designed but will be automatically designed through the use of the trust region optimization algorithm	The miniaturized microstrip antenna was printed and tested.	Section 4.2.2
The deliverable is the miniaturized microstrip antenna. The gain will not be directly designed but will be automatically designed through the use of the trust region optimization algorithm	The miniaturized microstrip antenna was printed and tested.	Section 4.2.3
The TRO algorithm that utilizes surrogate modeling. The aspects to be designed and implemented are the surrogate model, the optimization algorithm, the cost function, and the trust region algorithm	NA	The software is available on the elecotronic copy of the report and has also been added to the Technical Documentation
The TRO algorithm that utilizes surrogate modeling. The aspects to be designed and implemented are the surrogate model and the cost function	NA	The software is available on the elecotronic copy of the report and has also been added to the Technical Documentation

Table 3.
Summary of deliverables

4.2 Qualification tests

4.2.1 Qualification test 1: Measurement of the frequency spectrum of the miniaturized microstrip antenna

Objectives of the test or experiment

The objective of this test was to prove that the printed miniaturized microstrip antenna operated within the desired frequency range.

Equipment used

The following equipment was used:

- SMA connector
- 50 Ohm coaxial cable
- Network analyzer
- PC capable of interpreting the results of the network analyzer

Test setup and experimental parameters

The following steps were followed in order to setup the test:

1. The SMA connector was soldered with the center pin of the SMA connector connected to the feed of the printed microstrip antenna.
2. The printed microstrip was then connected to network analyzer via the 50 Ohm coaxial cable
3. The desired frequency range was setup through the use of the UI on the network analyzer
4. The test was executed through the use of the UI on the network analyzer

Results or measurements

The measured bandwidth of the printed microstrip antenna fell within the 978.8 MHz to 1015.7 MHz frequency. The resonant frequency was 997 MHz with return loss of $s_{11} = -27dBi$.

Observations

The frequency spectrum of the microstrip antenna was 77 MHz higher than it was expected to be. However, the bandwidth was very similar to the expected bandwidth.

4.2.2 Qualification test 2: Measurement of the gain of the miniaturized microstrip antenna

Objectives of the test or experiment

The objective of this test was to prove that the printed miniaturized microstrip antenna operated with a gain of $1.25dBi$ at the desired operating frequency.

Equipment used

The following equipment was used:

- SMA connector
- 50 Ohm coaxial cable
- Compact range and anechoic chamber

Test setup and experimental parameters

The following steps were followed in order to setup the test:

1. The SMA connector was soldered with the center pin of the SMA connector connected to the feed of the printed microstrip antenna.
2. The printed microstrip was then connected to compact range via the 50 Ohm coaxial cable. This was setup in the anechoic chamber.
3. The simulation was then setup in order to measure the gain of the printed microstrip antenna's E-plane and H-plane
4. The frequencies of the tests were changed from the original desired operating frequency of the printed microstrip antenna to perform 5 tests at 980 MHz, 990 MHz, 1.0 GHz, 1.01 GHz, and 1.02 GHz respectively

Steps followed in the test or experiment

Once the setup of the test was completed the test was executed via the use of the UI for the compact range. The compact range then automatically performed the test by rotating the printed microstrip antenna with respect to the transmitting antenna.

Results or measurements

The results of the test are then as follows revealed that the printed microstrip antenna had a maximum gain of 0.0 dBi at the its operating frequency.

Observations

The gain of the printed microstrip antenna is very similar to that of the simulations. It should be noted however that these values were for the operating frequency of the printed microstrip antenna as opposed to the desired operating frequency.

4.2.3 Qualification test 3: Analysis of the percentage area reduction of the miniaturized microstrip antenna

Results or measurements

The surface area of the baseline microstrip antenna geometry can be calculated through the use of the following Figures:

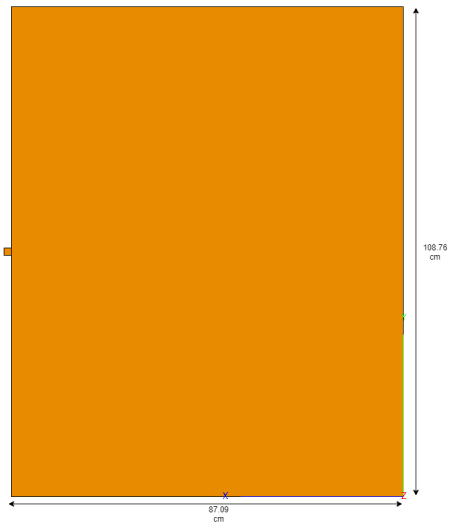


Figure 11.
Baseline Microstrip Antenna Ground Plane

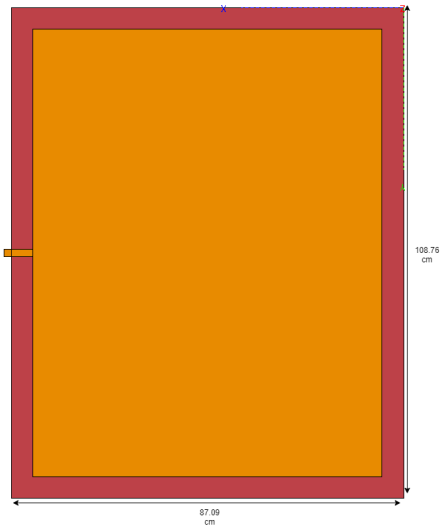


Figure 12.
Baseline Microstrip Antenna Radiating Plane

In Figure 11 and Figure 12 the baseline microstrip antenna's radiating and ground plane measurements are displayed. The baseline microstrip antenna then has the following parameters:

- Length: 87.08759 mm
- Width: 108.75646 mm

With a surface area of:

$$A_b = 87.08759mm * 108.75646mm \tag{48}$$

$$A_b = 9471.33mm^2 \tag{49}$$

The surface area of the printed microstrip antenna geometry can be calculated through the use of Figure 13 and Figure 14.

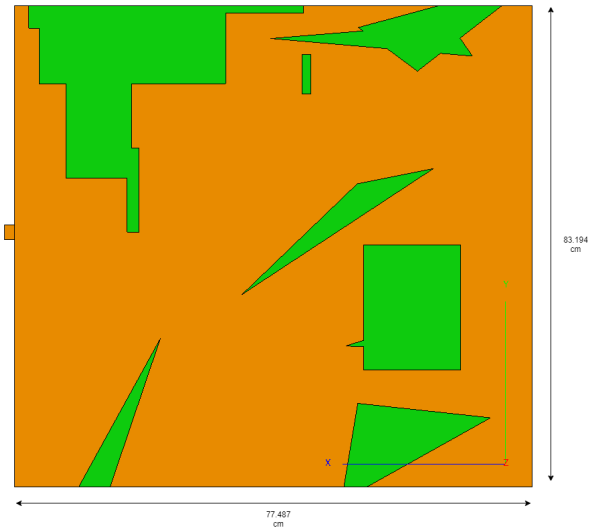


Figure 13.
Miniaturized Microstrip Antenna Ground Plane

In Figure 13 and Figure 14 the miniaturized microstrip antenna's radiating and ground plane measurements are displayed. The baseline microstrip antenna then has the following parameters:

- Length: 77.487 mm
- Width: 83.184 mm

With a surface area of:

$$A_m = 83.184mm * 77.487mm \tag{50}$$

$$A_m = 6446.45mm^2 \tag{51}$$

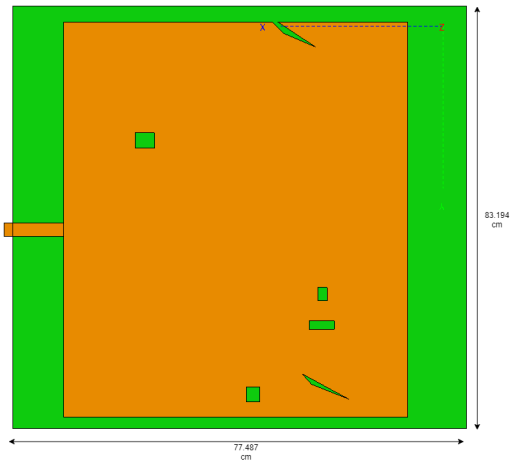


Figure 14.
Miniaturized Microstrip Antenna Radiating Plane

Observations

The required area reduction as specified in Table 2 is 45%. The achieved percentage area reduction is:

$$A_{reduction} = (\quad 1.0 \quad - \quad \frac{A_m}{A_b} \quad) * 100.0\% \tag{52}$$

$$A_{reduction} = (\quad 1.0 \quad - \quad \frac{6446.45mm^2}{9471.33mm^2} \quad) * 100.0\% \tag{53}$$

$$A_{reduction} = 31.93\% \tag{54}$$

While it can be seen that the surface area of the microstrip antenna was reduced by nearly a third the 45% area reduction was not fully met.

4.2.4 Qualification test 4: Analysis of the amount of simulations that the software solution required to optimize the baseline microstrip antenna

Results or measurements

The total amount of low-fidelity simulations performed was 400. This is due to the fact that 8 low-fidelity simulations were performed per iteration of the TRO algorithm.

Observations

Whilst the number of low-fidelity simulations performed greatly exceeds the number of high-fidelity simulations performed, these two parameters should not be directly compared with each other. The intended outcomes found in Table 2 specifies that within the period of 3 hours 45 high-fidelity simulations could be performed. This means that a single high-fidelity simulation

could not take more than 4 minutes.

The low-fidelity simulations each took approximately 30 seconds, or in other words 8 low-fidelity simulations could be performed within a period of 4 minutes. For this reason the number of low-fidelity simulations that were run per iteration of TRO algorithm was set to 8. The total number of iterations of the TRO algorithm was 50 when the exit condition was reached for the TRO algorithm. The 400 low-fidelity simulations that were run during the execution of the TRO algorithm is therefore equivalent to 50 high-fidelity simulations in terms of execution time.

4.2.5 Qualification test 5: Analysis of the amount of time that the software solution required to optimize the baseline microstrip antenna

Observations

Through the use of a primitive UI implemented via command prompt the execution time of the TRO algorithm could be measured. The full execution time for the TRO algorithm was then 4:44.962 . In other words the TRO algorithm executed in approximately 4.7 hours, this is $\frac{4.7}{3} = 1.56$ times more than the intended outcome.

5. Discussion

5.1 Interpretation of Results

Bandwidth

The very first and most import result to take note of is the frequency spectrum of the printed microstrip antenna. When we look at the results obtained in Section 4.2.1 it is very clear that the operating frequency of the printed microstrip antenna does not match the frequency spectrum of the simulation that was performed to obtain said printed microstrip antenna.

The shift in the operating frequency of the printed microstrip antenna could be due to the following:

- During the printing process one of the many slots in the ground plane of the miniaturized microstrip antenna was printed slightly wrong which in turn caused a frequency shift
- The lack of high-fidelity simulations in the implementation of the TRO algorithm

The above points are further discussed in Section 5.2.

Gain

We now turn our attention to the results in Section 4.2.2. It is clear to that we did not meet the gain requirement set for the miniaturized microstrip antenna. Whilst the gain of the printed microstrip antenna does not completely meet the desired specifications it is still an improvement over the -3.0 dBi at 900 MHz and 920 MHz and -1.5 dBi at 940 MHz of the baseline microstrip antenna.

Whilst the gain of microstrip antennas does tend to be low the gain specification should not be out of reach as it is not a large gain value.

Percentage Miniaturization

The amount of miniaturization achieved by the TRO algorithm during this specific execution does not meet the desired miniaturization. Whilst the reason for this might not be clear in hindsight observing the execution of the TRO algorithm in person would have made the reason clear. That is, the TRO algorithm became stuck in a local minimum where the amount of miniaturization was decreased due to the fact that a very low frequency fitness was achieved at the specific point in the objective function. Due to the implementation of the fitness function in Equation 47 any change in the geometry of the antenna then caused an increase in the the fitness of the frequency for the antenna geometry which caused the overall fitness of the candidate antenna geometry to increase.

5.2 Critical Evaluation of the Design

Surrogate Model

In terms of flexibility and functionality the implementation of the ANN as the surrogate model is considered to have been successful. However, in hindsight it becomes clear that a Kriging model should have been used as the surrogate model for this project as opposed to an ANN. The reason for this being that Kriging models tend to achieve higher precision and are more effective in problems with high dimensionality than the ANN. Alternatively, the ANN could have been implemented to use more complex training methods such as those in Section 3.1.4 which could perhaps have allowed the ANN surrogate model to more accurately map the objective function given the small data set.

Additionally, hyper-parameter optimization could have improved the performance of the ANN as well. However, due to time limitations this could not be implemented.

Multivariate Optimization Algorithms

The implemented TRO and PSO optimization algorithms had a somewhat mediocre performance with regard to mapping the objective function, however they performed well when used as part of the training method of the surrogate model. The reason they performed poorly on the mapping of the objective function is likely due to the fact the two algorithms were not confined to the trust region. That is, through the application of momentum in the case of PSO and the inherent qualities of the TRO algorithm to move away from the original search space the two algorithms would tend to map candidates outside of the trust region. These candidates would then clearly not be as accurate as those with in the trust region due to the fact that the surrogate model would have little to no data for areas outside of the trust region.

In terms of execution time both algorithms usually finished their problems in 3 seconds or less on a single CPU core.

Antenna Optimization Handler

When taking the AOH implementation into consideration several design flaws come to mind. The first and probably most prominent of these flaws occurred in the Section 2.3. That is, the overall approach of the implementation of the software solution would never have achieved the desired specifications. The major approach flaw that was made that affected this was the choice to utilize low-fidelity simulations at run time instead of for instance using a data base of candidate antenna geometries that have been previously simulated. The choice to use only randomly generated candidate antenna geometries during the execution of the TRO algorithm caused the data set used to train the surrogate to be much smaller than it would have been. With a larger data set of candidate antenna geometries that have been previously simulated the surrogate would have more training data and therefore would be likely to be more accurate.

The second major flaw in the approach to the implementation of the problem is the distinct lack of high-fidelity simulations. Due to the fact that high-fidelity simulations are more accurate they should, at the very least, have been used to verify that new center candidate antenna geometries would behave as expected in a real-world scenario. As mentioned in Section 5.1 it is suspected that this is the cause for the difference in the simulated and measured operating frequencies of the miniaturized microstrip antenna. This can be easily verified by increasing the fidelity of the

original simulation used for the printed microstrip antenna.

Finally, the parameters c_1, c_2, and c_3 used in the fitness function implemented in Equation 47 have an extremely large impact on the overall fitness of the candidates antenna geometries. Due to this fact it is clear that whilst due to time limitations most of the parameters used in the software solution could not be optimized, these three scalar values should have been optimized. Had they been optimized it is very likely that the TRO algorithm could've achieved a better miniaturized antenna geometry and would not have gotten stuck in a local minimum.

Software Solution

If the entirety of the software solution is then taken into consideration, the following comes to mind:

- Whilst flexibility of the solution was not a specification it was none the less an extremely important part of the project due to the nature of the problem. However, flexibility should not have out-weighed the functionality required when the approach was being considered.

5.3 Design ergonomics

The software solution was designed to both be flexible and easy to use. This was accomplished through the use of configuration files that contain the initialization parameters of the functional units. By altering these configuration files it is then easy to alter the functionality of the functional units.

A primitive UI was also implemented in order to give the user a better perspective of how the TRO algorithm is performing at any point in time.

6. Conclusion

6.1 Summary of the Work Completed

In this project a Trust-Region Optimization algorithm was developed that utilizes two phases to optimize a baseline microstrip antenna geometry. In the primary phase of the optimization algorithm the baseline microstrip antenna geometry is optimized through the introduction of rectangular, triangular, elliptical, and polygonal slots to both the radiating plane and the ground plane. In the secondary phase a subset of parameters is chosen to be optimized at a time. This subset of parameters is then optimized through the use of surrogate modelling where the mapping of the surrogate is accomplished through the use of a multi-process Trust-Region Optimization and Particle Swarm Optimization algorithm. All the candidates generated during the execution of the TRO algorithm are simulated through the use of the Feko API and Lua scripting. Low-fidelity simulations of the candidate antenna geometries are then performed in order to save time due to the data acquisition method used in this approach.

6.2 Summary of the observations and findings

The use of only low-fidelity simulations during the TRO algorithm proved detrimental due to the fact that there was a large frequency shift in the simulation of the final microstrip and the obtained results.

6.3 Contribution

The following work was new to the author:

- Multivariate Optimization Algorithms: TRO, PSO, Nelder-Mead, and Bee-Colony Optimization
- Surrogate Modelling: Kriging and SVMr
- Artificial Neural Network: The in depth implementation of the ANN
- Microstrip Antenna Miniaturization
- Microstrip Antenna Miniaturization through the introduction to slots to the ground plane
- The Feko API and Lua scripting language were mastered

6.4 Future Work

The following functionality should be implemented should work continue on this project in the future:

- A means of validating candidate antenna geometries through the use of high-fidelity simulations
- The creation of a data set of simulated antenna geometries where the data set should be expanded each time a new simulation is run
- Hyper-parameter optimization of the scalars used in the fitness function
- Alternative training methods should be implemented for the ANN surrogate model
- A Kriging model should be implemented as an additional surrogate model
- Surrogate models implemented in public libraries should be used to validate the performance of the surrogate model used
- Additional multivariate optimization algorithms can be implemented for surrogate mapping
- An advances Bee-Colony Optimization algorithm developed by the author can be tested

7. References

[1] J. N. D. O. "R.A. Waltz", "J.L. Morales', "An interior algorithm for nonlinear optimization that combines line search and trust region steps," https://link.springer.com/article/10.1007/s10107-004-0560-5, July 2006, accessed on 2019-10-31.

[2] A. G. "P.C. Fourie", "The particle swarm optimization algorithm in size and shape optimization," https://link.springer.com/article/10.1007/s00158-002-0188-0citeas, May 2002, accessed on 2019-10-31.

[3] Y. C. "S.K. Behera", "Design and optimization of dual band microstripantenna using particle swarm optimization technique," https://link-springer-com.uplib.idm.oclc.org/content/pdf/10.1007%2Fs10762-010-9722-0.pdf, Sept 2010, accessed on 2019-10-31.

[4] A. B. "S. Koziel", "Fast simulation-driven feature-based designoptimization of compact dual-band microstripbranch-line coupler," https://onlinelibrary.wiley.com/doi/epdf/10.1002/mmce.20923, April 2015, accessed on 2019-10-31.

[5] Z. Y. X. Z. W. W. "J.B. Jiang", "Y. Song", "Band-notched uwb printed antenna with an inverted-l-slotted ground," https://onlinelibrary.wiley.com/doi/epdf/10.1002/mop.24003, May 2008, accessed on 2019-10-31.

[6] M. K. "A.E. Yilmaz", "Calculation of optimized parameters of rectangular microstrip patch antenna using particle swarm optimization," https://onlinelibrary-wiley-com.uplib.idm.oclc.org/doi/epdf/10.1002/mop.22918, May 2007, accessed on 2019-10-31.

[7] C. A. Balanis, *Antenna theory: analysis and design*. John wiley & sons, 2016.

[8] A. Bekasiewicz and S. Koziel, "Cost-efficient design optimization of compact patch antennas with improved bandwidth," *IEEE Antennas and Wireless Propagation Letters*, vol. 15, pp. 270–273, 2015.

[9] Feko, "Feko user manual," https://www.altairuniversity.com/wp-content/uploads/2015/03/UserManual.pdf, May 2014, accessed on 2019-10-31.

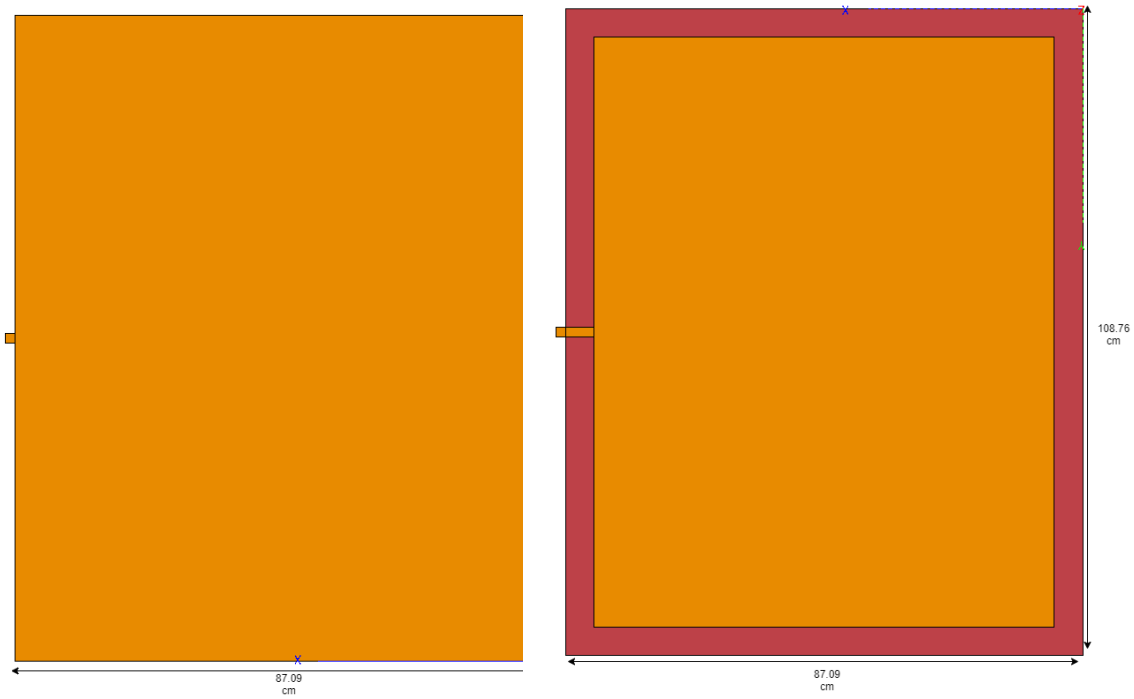
1. Appendix A: Additional Simulation Results

In order for the miniaturized microstrip antenna to be put into context the baseline microstrip antenna's results will first be covered.

1.1 Baseline Microstrip Antenna

1.1.1 Geometry

The baseline microstrip antenna geometry, designed from the equations provided in Balanis, is as follows:



(a) Baseline Microstrip Antenna Ground Plane (b) Baseline Microstrip Antenna Radidating Plane

Figure 1

Above, in Figure 2, the baseline microstrip antenna radiating and ground planes are displayed. The baseline microstrip antenna has an area of:

$$Area_b = 87.08759mm * 108.75646mm \tag{1.1}$$

$$Area_b = 9471.33mm^2 \tag{1.2}$$

Below a 3D representation of the baseline microstrip is given should a better understanding of the geometry be required.

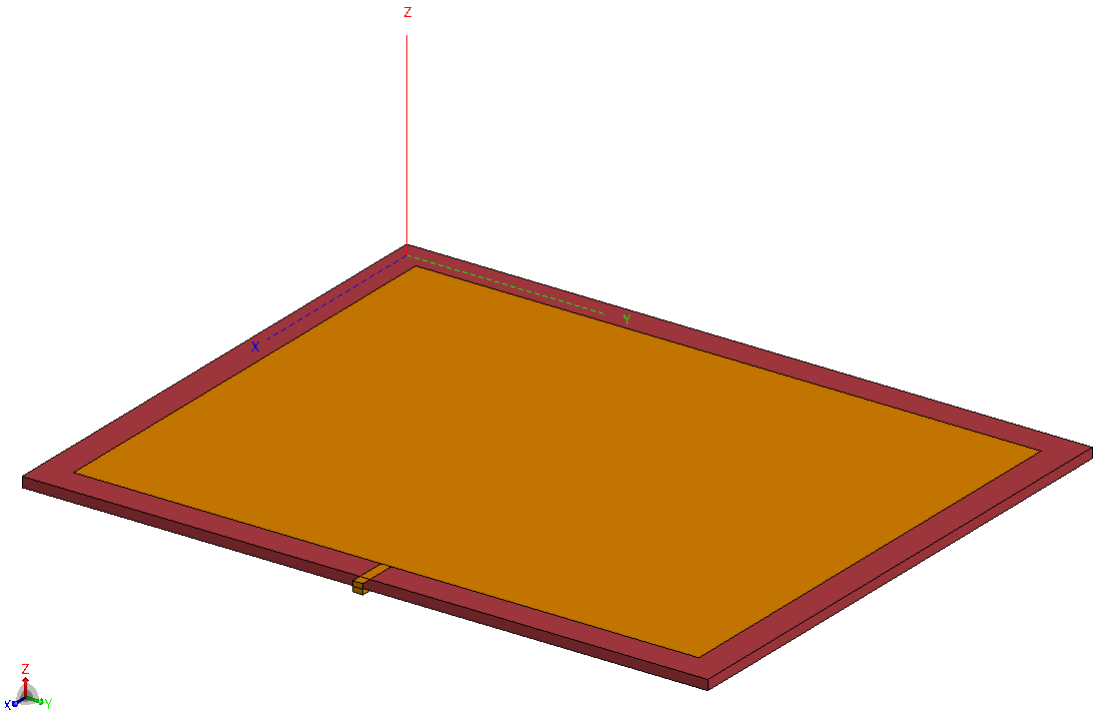


Figure 2: Baseline Microstrip Antenna 3D View

1.1.2 Frequency Response

The frequency response for the baseline microstrip is shown in Figure 4 on the following below. From Figure 4 the following important information can be seen:

- BW: 15.5812 MHz
- Lower Edge Frequency, F_{lower}: 916 MHz
- Upper Edge Frequency, F_{upper}: 931.58 MHz
- Center Frequency, F_{center}: 924 MHz

The fractional bandwidth of the baseline microstrip antenna is then calculated as follows:

$$FBW = \frac{F_{upper} - F_{lower}}{F_{center}} \tag{1.3}$$

$$FBW = \frac{931.58 - 916}{924} \tag{1.4}$$

$$FBW = 16.861 \times 10^{-3} \tag{1.5}$$

$$FBW = 1.686\% \tag{1.6}$$

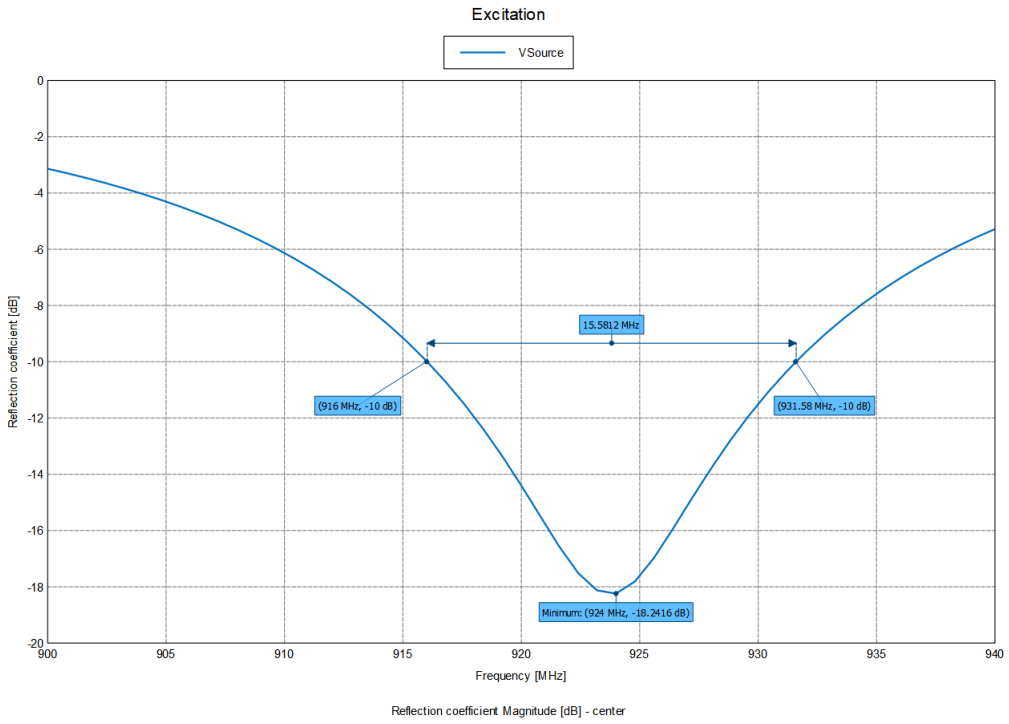
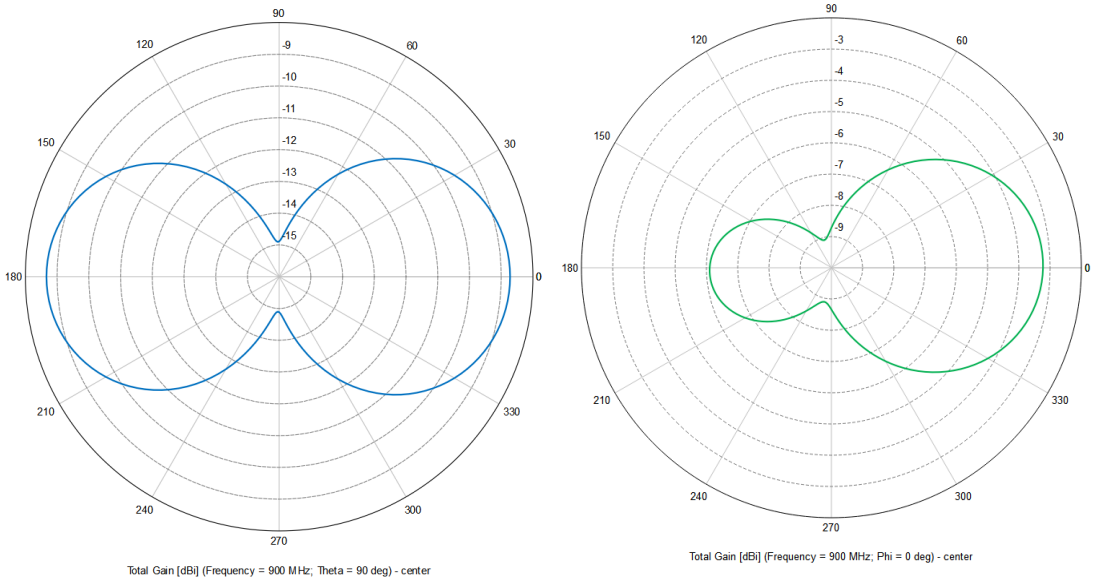


Figure 3: Baseline Microstrip Antenna Frequency Response

1.1.3 EM Response

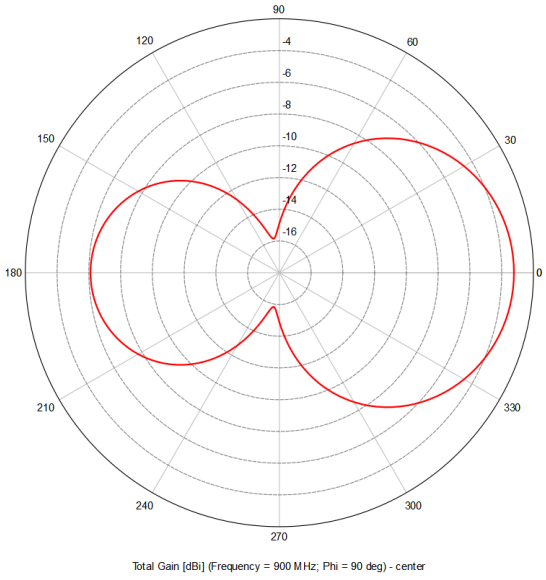
The baseline microstrip antenna then has the following EM response:



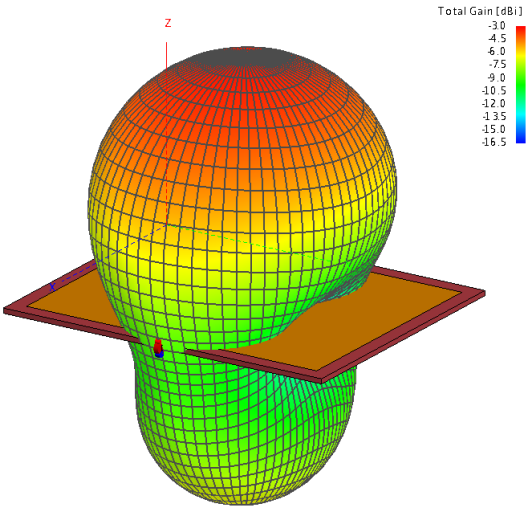
(a) Horizontal Plane at 900 MHz

(b) E-Plane at 900 MHz

Figure 4

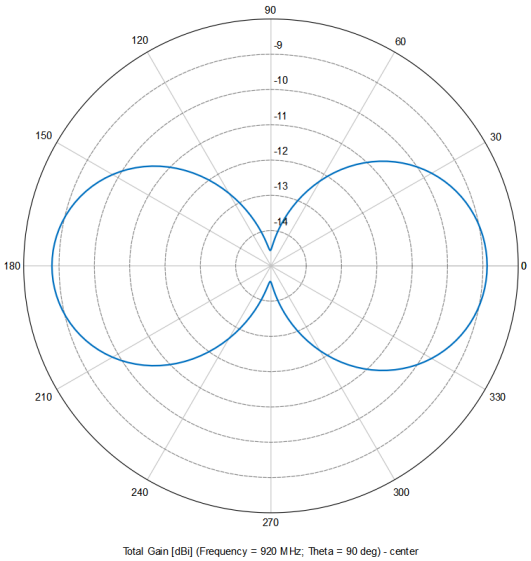


(a) H-Plane at 900 MHz

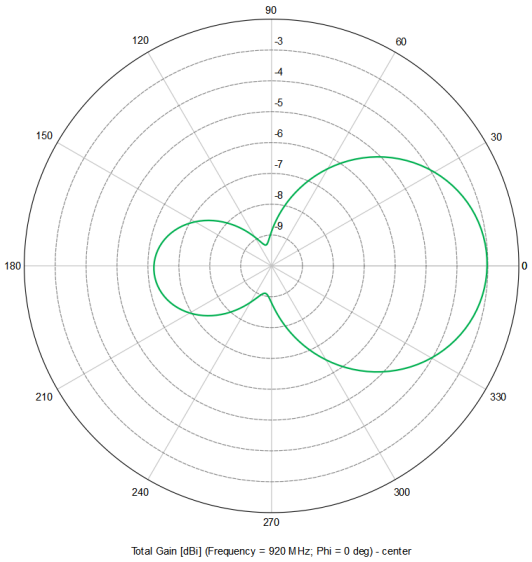


(b) 3D EM Response at 900 MHz

Figure 5

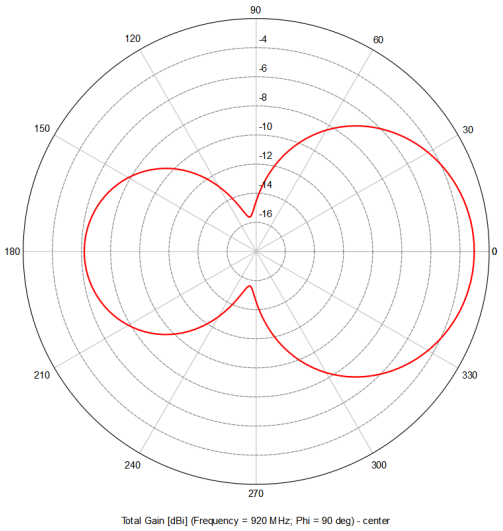


(a) Horizontal Plane at 920 MHz

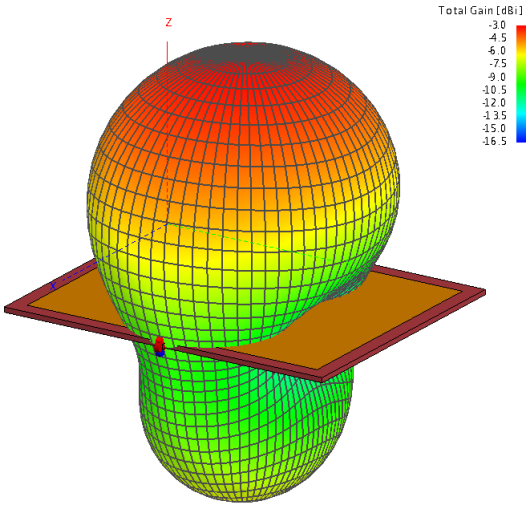


(b) E-Plane at 920 MHz

Figure 6

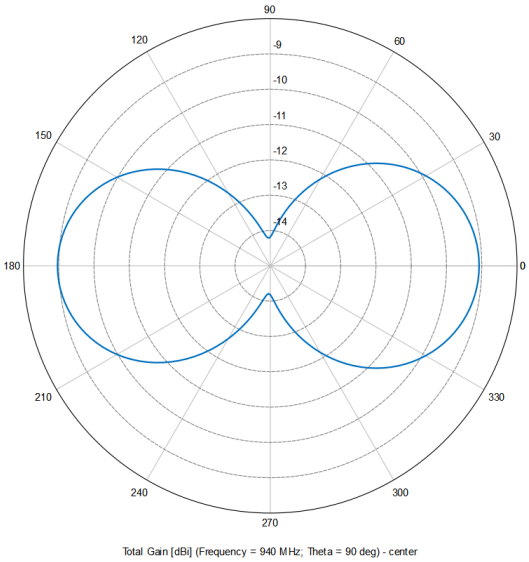


(a) H-Plane at 920 MHz

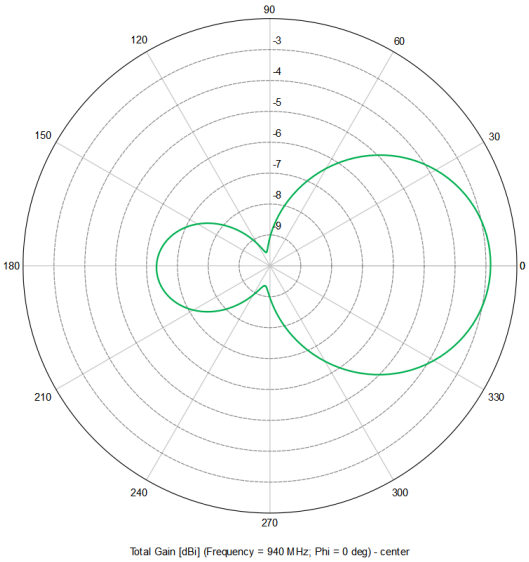


(b) 3D EM Response at 920 MHz

Figure 7

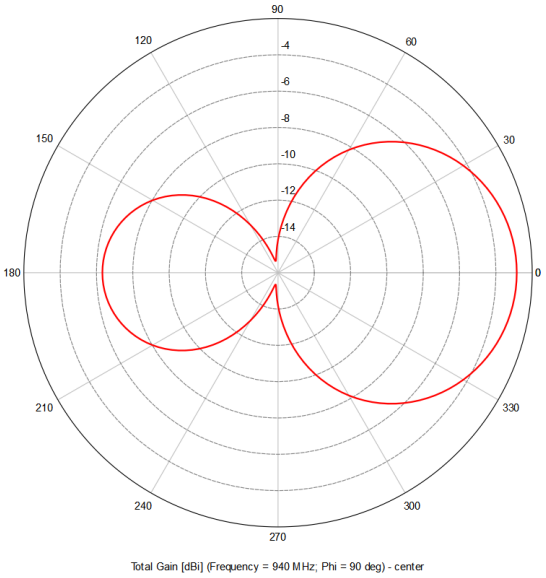


(a) Horizontal Plane at 940 MHz

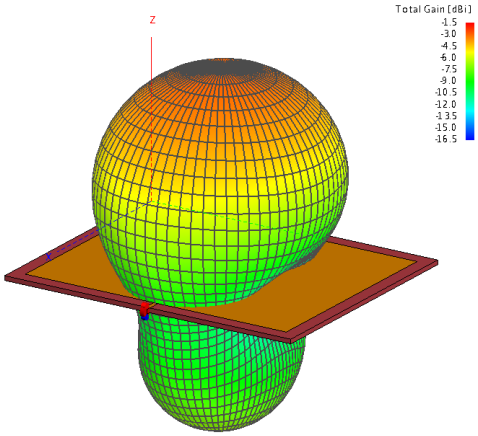


(b) E-Plane at 940 MHz

Figure 8



(a) H-Plane at 940 MHz



(b) 3D EM Response at 940 MHz

Figure 9

From Figure 5 through to Figure 10 we can then see that the gain of the baseline microstrip antenna is as follows:

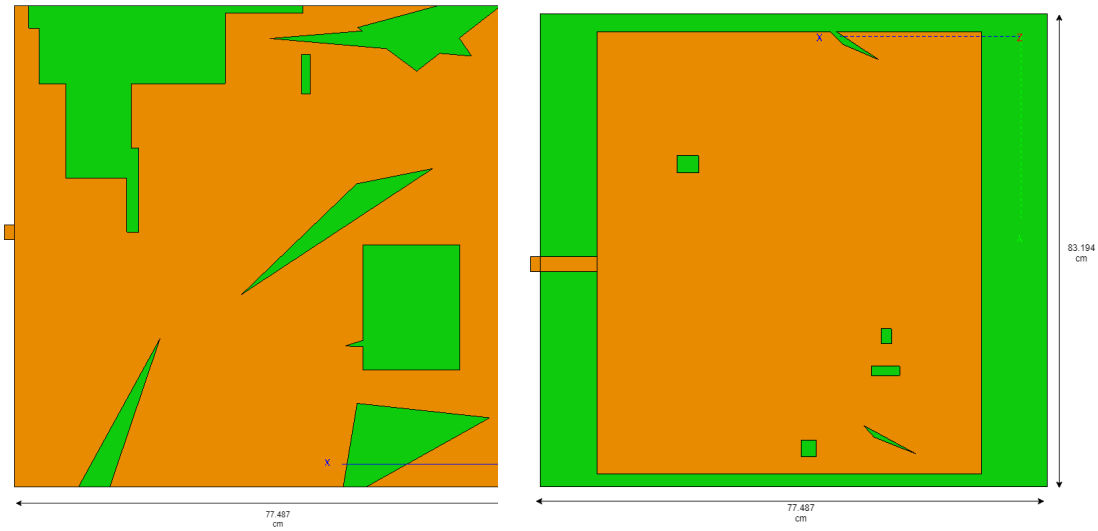
- 900 MHz: -3.0 dBi
- 920 MHz: -3.0 dBi
- 940 MHz: -1.5 dBi

It can be noted that this value is already below the 1.25 dBi objective.

1.2 Miniaturizes Microstrip Antenna

1.2.1 Geometry

The miniaturized microstrip antenna geometry is as follows:



(a) Miniaturized Microstrip Antenna Ground Plane

(b) Miniaturized Microstrip Antenna Radiating Plane

Figure 10

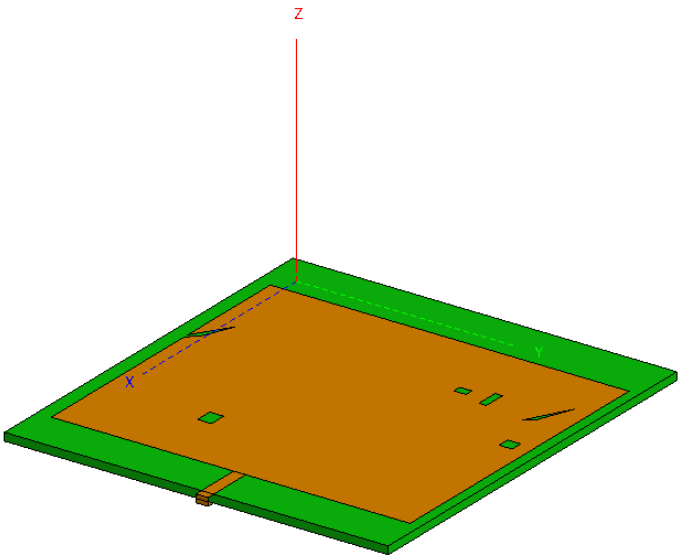


Figure 11: Miniaturized Microstrip Antenna 3D View

The area for the miniaturized microstrip antenna is then:

$$Area_m = 83.184mm * 77.487mm \tag{1.7}$$

$$Area_m = 6446.45mm^2 \tag{1.8}$$

Below a 3D representation of the miniaturized microstrip is given in Figure 12 should a better understanding of the geometry be required.

1.2.2 Frequency Response

The frequency response for the miniaturized mircrostrip antenna is shown below in Figure 13.

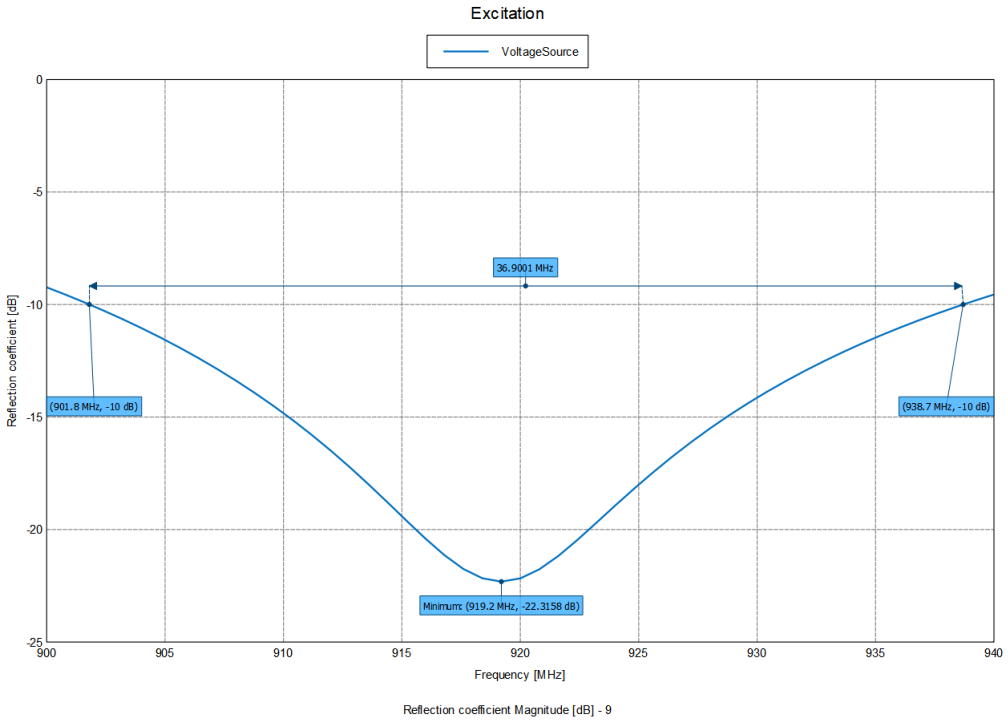


Figure 12: Miniaturized Microstrip Antenna Frequency Response

The following information can be found in Figure 13:

- BW = 36.9 MHz
- Lower Edge Frequency, F_{lower}: 901.8 MHz
- Upper Edge Frequency, F_{upper}: 938.7 MHz
- Center Frequency, F_{center}: 919.2 MHz

The fractional bandwidth for the miniaturized microstrip antenna is then:

$$FBW = \frac{F_{upper} - F_{lower}}{F_{center}} \tag{1.9}$$

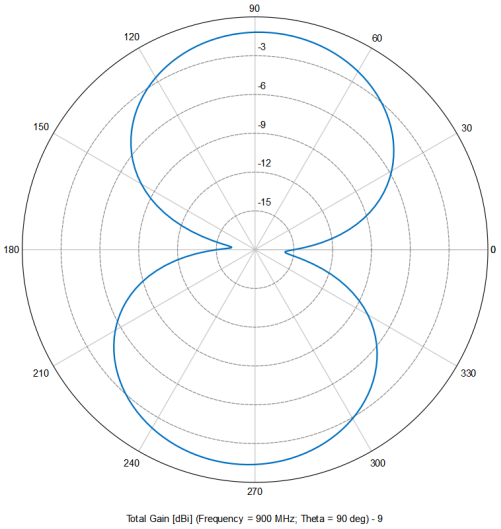
$$FBW = \frac{938.7 - 901.8}{919.2} \tag{1.10}$$

$$FBW = 40.14 \times 10^{-3} \tag{1.11}$$

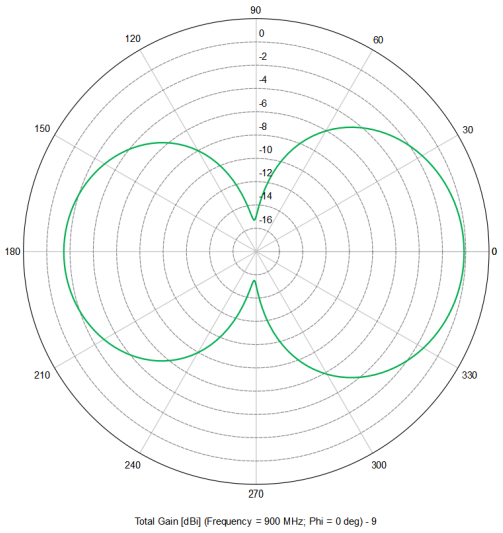
$$FBW = 4.0014\% \tag{1.12}$$

1.2.3 EM Response

The miniaturized microstrip antenna then has the following EM response:

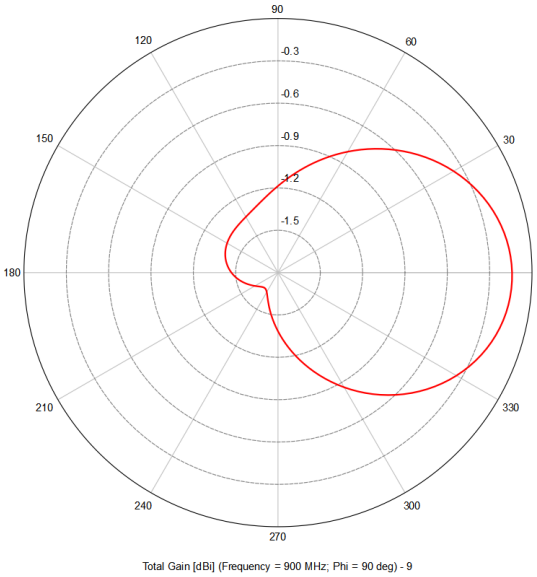


(a) Horizontal Plane at 900 MHz

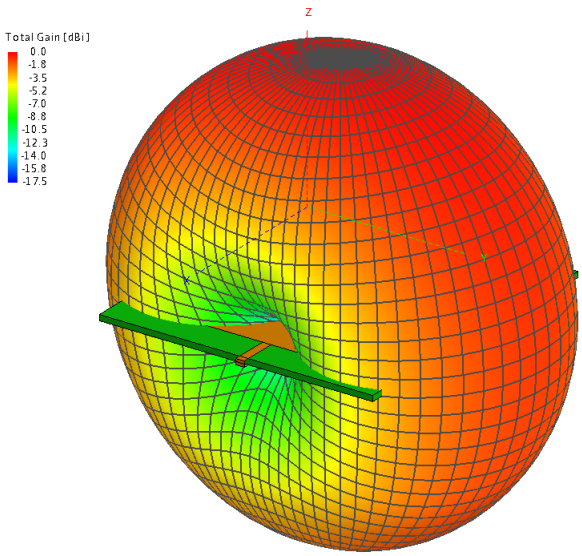


(b) E-Plane at 900 MHz

Figure 13

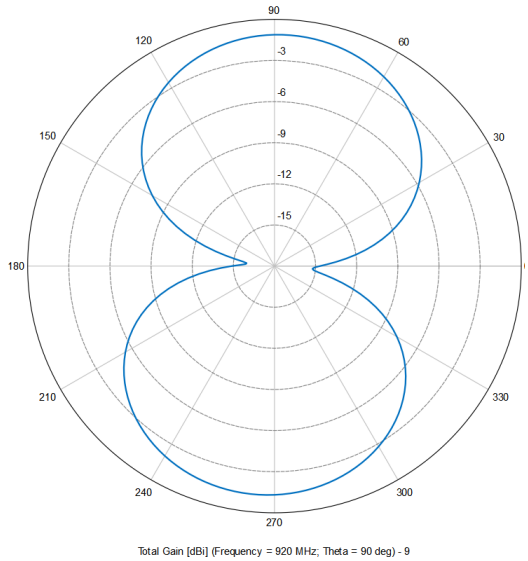


(a) H-Plane at 900 MHz

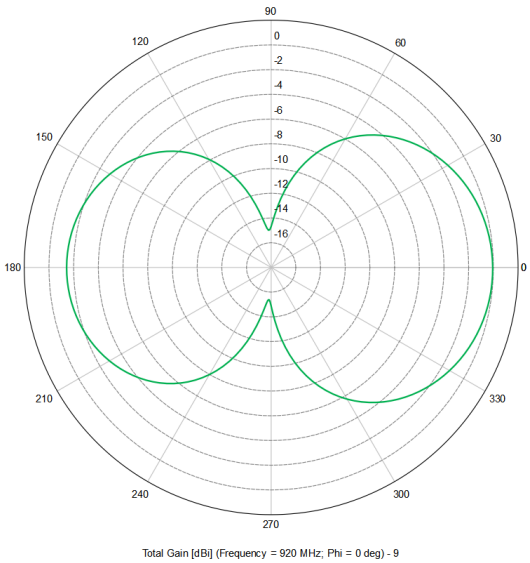


(b) 3D EM Response at 900 MHz

Figure 14

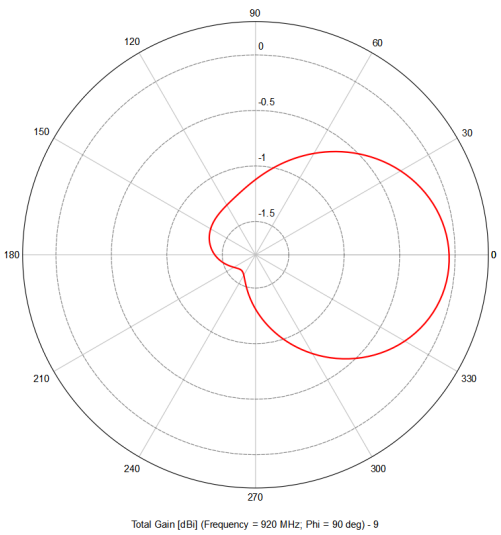


(a) Horizontal Plane at 920 MHz

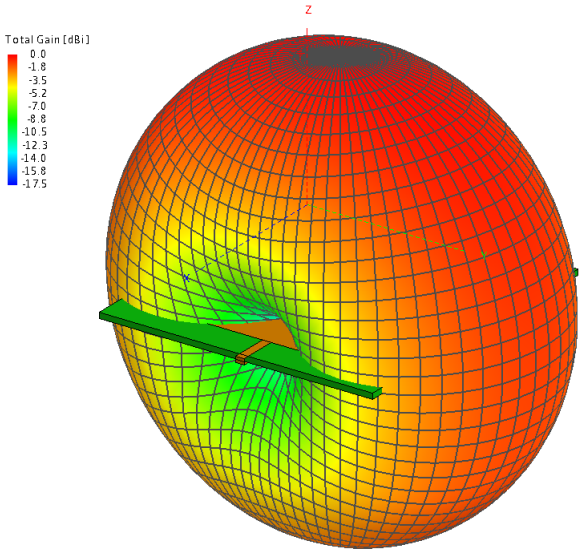


(b) E-Plane at 920 MHz

Figure 15

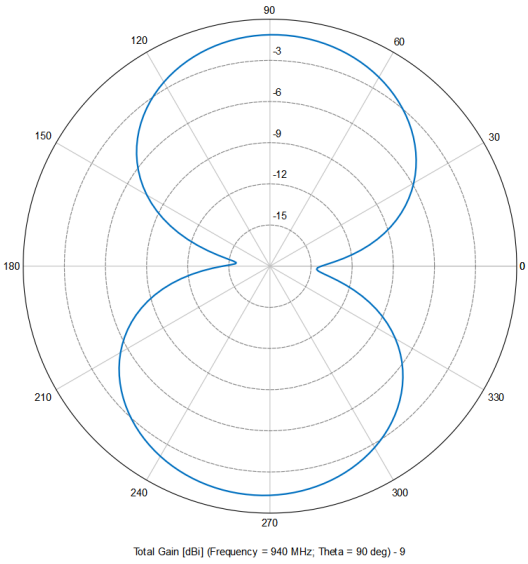


(a) H-Plane at 920 MHz

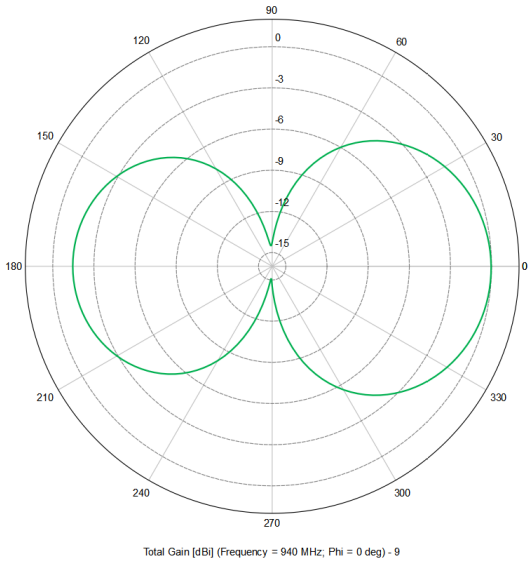


(b) 3D EM Response at 920 MHz

Figure 16

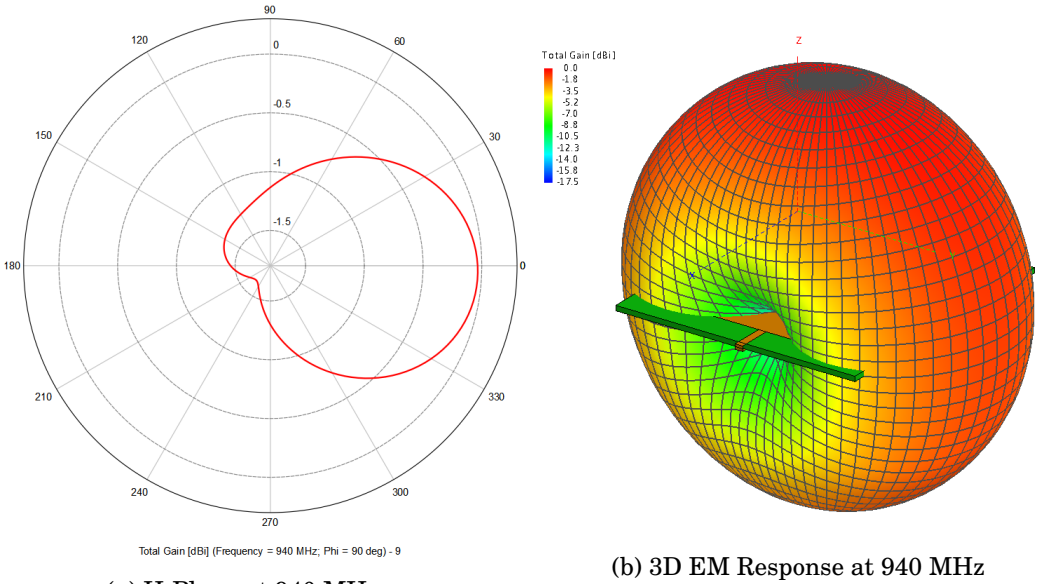


(a) Horizontal Plane at 940 MHz



(b) E-Plane at 940 MHz

Figure 17



(a) H-Plane at 940 MHz

(b) 3D EM Response at 940 MHz

Figure 18

From Figure 14 through to Figure 19 we can then see that the gain of the miniaturized microstrip antenna is as follows:

- 900 MHz: 0.0 dBi
- 920 MHz: 0.0 dBi
- 940 MHz: 0.0 dBi

1.2.4 Slots

The miniaturized microstrip antenna then has the following unique slots in it's radiating plane and ground plane respectively:

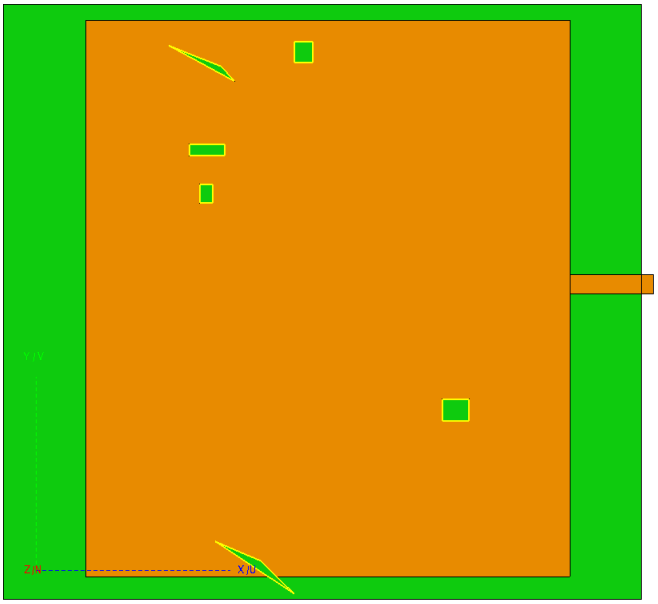


Figure 19: Miniaturized Microstrip Antenna Radiating Plane Slots

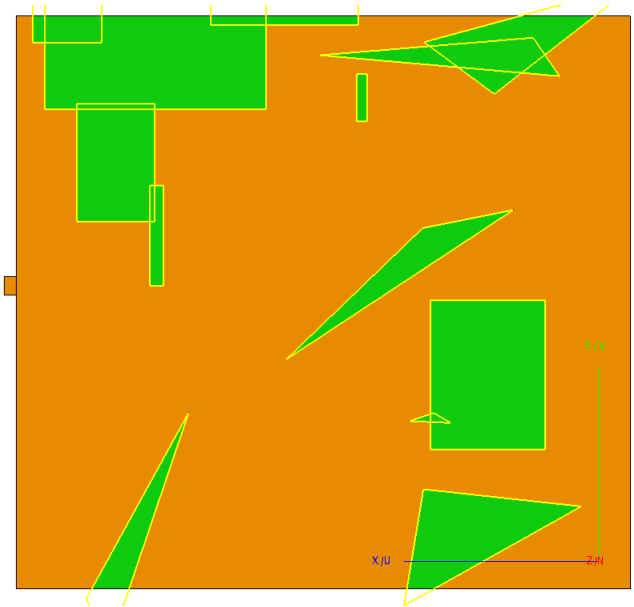


Figure 20: Miniaturized Microstrip Antenna Ground Plane Slots

As can be seen in Figure 20 and Figure 21 both the radiating plane and the ground plane of the miniaturized microstrip antenna have multiple unique slots, with the ground plane having many more than radiating plane. The radiating plane contains 6 unique slots whereas the ground plane contains 13 unique slots.

As was mentioned in Section 4.1 each antenna is characterised by a dictionary that contains all its parameters. The dictionary for this antenna can be found in Appendix A.

1.3 Surrogate Model

The surrogate model being used for the optimization of the microstrip antenna is a fully-connected neural network. The neural network has the following features:

- Learning Rate: 0.05
- Momentum: 0.001
- Number of Input Nodes: 30
- Hidden Layers: 2
- Hidden Layer Width: Layer 1 = 36, Layer 2 = 35
- Number of Output Nodes: 1
- Regularization: L1 Regularization
- Accuracy: 72%
- Training Method: Stochastic gradient decent with early stopping for over-fitting
- Activation Function: tanH
- Bias Nodes: Default bias node values are 1.0, with the weights being updated as normal weights

After training the neural network was as follows:

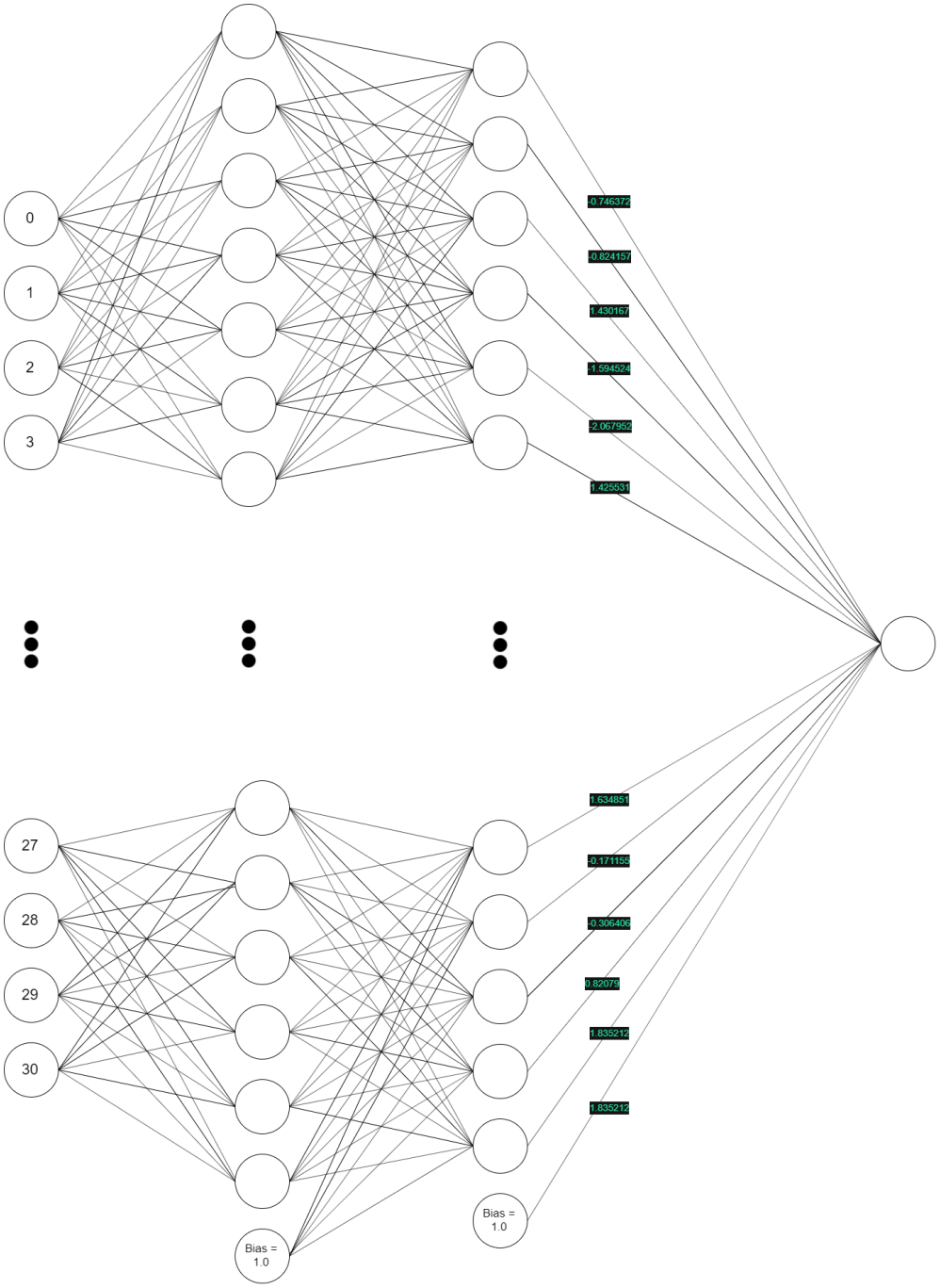


Figure 21: Shallow Fully Connected Artificial Neural Network

Due to how wide the data is there are 1050 weights for the first layer of weights alone which makes a visualization of the data for the neural network nearly impossible. The general geometry of the neural network is shown in Figure 22 along with the final layer of weights for the nodes that are being shown. The full data for the neural network can be found in the attached "neural net.json" file, which can be opened with a normal text editor.

The input parameters for the neural network were the following:

1. Microstrip Feed Center Position
2. Microstrip Feed Width
3. Ground Plane Length
4. Ground Plane Width
5. Ground Plane X-coordinate
6. Ground Plane Y-coordinate
7. Radiating Plane Length
8. Radiating Plane Width
9. Radiating Plane X-coordinate
10. Radiating Plane Y-coordinate
11. Ground Plane Rectangular Slot 2: Length
12. Ground Plane Rectangular Slot 2: Width
13. Ground Plane Rectangular Slot 2: X-coordinate
14. Ground Plane Rectangular Slot 2: Y-coordinate
15. Ground Plane Triangular Slot 3: Corner 0, X-coordinate
16. Ground Plane Triangular Slot 3: Corner 0, Y-coordinate
17. Ground Plane Triangular Slot 3: Corner 1, X-coordinate
18. Ground Plane Triangular Slot 3: Corner 1, Y-coordinate
19. Ground Plane Triangular Slot 3: Corner 2, X-coordinate
20. Ground Plane Triangular Slot 3: Corner 2, Y-coordinate
21. Radiating Plane Rectangular Slot 0: Length
22. Radiating Plane Rectangular Slot 0: Width
23. Radiating Plane Rectangular Slot 0: X-coordinate
24. Radiating Plane Rectangular Slot 0: Y-coordinate
25. Radiating Plane Triangular Slot 0: Corner 0, X-coordinate
26. Radiating Plane Triangular Slot 0: Corner 0, Y-coordinate
27. Radiating Plane Triangular Slot 0: Corner 1, X-coordinate
28. Radiating Plane Triangular Slot 0: Corner 1, Y-coordinate
29. Radiating Plane Triangular Slot 0: Corner 2, X-coordinate
30. Radiating Plane Triangular Slot 0: Corner 2, Y-coordinate

While the outputs for the neural network were the following:

1. The overall fitness of the candidate antenna geometry. The method for obtaining the fitness is discussed in Section 4.2, Equation 6.41 .

2. Appendix B: Additional Simulation Conclusions

2.1 Miniaturization

The required area reduction as specified in Section 1.1 was 45%. The achieved percentage area reduction is:

$$Area_{reduction} = (\quad 1.0 \quad - \quad \frac{A_m}{A_b} \quad) * 100.0\% \tag{2.13}$$

$$Area_{reduction} = (\quad 1.0 \quad - \quad \frac{6446.45mm^2}{9471.33mm^2} \quad) * 100.0\% \tag{2.14}$$

$$Area_{reduction} = 31.93\% \tag{2.15}$$

While it can be seen that the surface area of the microstrip antenna was reduced by nearly a third the 45% area reduction was not fully met.

2.2 Bandwidth

The required bandwidth of the miniaturized microstrip antenna as specified in Section 1.1 is 40 MHz, with the achieved bandwidth is 36.9 MHz. Whilst this does not completely meet the desired specifications it is still a large improvement over the 15.5812 MHz bandwidth of the baseline microstrip antenna.

2.3 Gain

The required gain of the miniaturized microstrip antenna as specified in Section 1.1 is 1.25 dBi, with the measured gain being 0.0 dBi at 900 Mhz, 920 MHz, and 940 MHz. Whilst this does not completely meet the desired specifications it is still an improvement over the -3.0 dBi at 900 MHz and 920 MHz and -1.5 dBi at 940 MHz of the baseline microstrip antenna.

2.4 Surrogate

Considering the width of the data set 72% accuracy is quite good. The surrogate was also trained in approximately 2 minutes which is decent.

3. Appendix C: Final miniaturized antenna geometry parameters


```
"geometry" :
{
    "feed" :
    {
        "center" : 37.275,
        "width" : 2.444
    },
    "ground plane" :
    {
        "l" : 83.194,
        "w" : 77.487,
        "x" : -4.291,
        "y" : -3.725,
        "slots" :
        {
            "elliptical" :
            {
                "items" : 0
            },
            "rectangular" :
            {
                "items" : 8,
                "0" :
                {
                    "x" : 7.218,
                    "y" : 15.097,
                    "z" : 0.0,
                    "l" : 15.565,
                    "w" : 20.174
                },
                "1" :
                {
                    "x" : 58.878,
                    "y" : 37.308,
                    "z" : 0.0,
                    "l" : 1.94,
                    "w" : 13.501
                },
                "2" :
                {
                    "x" : 96.399,
                    "y" : 81.529,
                    "z" : 0.0,
                    "l" : 12.566,
                    "w" : 37.262
                },
                "3" :
```

```
        {
            "x" : 44.99,
            "y" : 61.172,
            "z" : 0.0,
            "l" : 30.012,
            "w" : 17.185
        },
        "4" :
        {
            "x" : 31.311,
            "y" : 59.504,
            "z" : 0.0,
            "l" : 1.414,
            "w" : 6.408
        },
        "5" :
        {
            "x" : 32.492,
            "y" : 72.554,
            "z" : 0.0,
            "l" : 20.075,
            "w" : 20.521
        },
        "6" :
        {
            "x" : 67.327,
            "y" : 70.092,
            "z" : 0.0,
            "l" : 9.303,
            "w" : 15.669
        },
        "7" :
        {
            "x" : 60.158,
            "y" : 45.94,
            "z" : 0.0,
            "l" : 10.475,
            "w" : 15.926
        }
    },
    "triangular" :
    {
        "items" : 6,
        "0" :
        {
            "items" : 3,
            "0" : {
                "x" : 23.822,
                "y" : 45.059,
                "z" : 0.0
```

```
        },
        "1" :
        {
            "x" : 42.349,
            "y" : 27.256,
            "z" : 0.0
        },
        "2" :
        {
            "x" : 11.546,
            "y" : 47.521,
            "z" : 0.0
        }
    },
    "1" :
    {
        "items" : 3,
        "0" :
        {
            "x" : 55.494,
            "y" : 20.14,
            "z" : 0.0
        },
        "1" :
        {
            "x" : 66.551,
            "y" : -12.516,
            "z" : 0.0
        },
        "2" :
        {
            "x" : 69.326,
            "y" : -4.997,
            "z" : 0.0
        }
    },
    "2" :
    {
        "items" : 3,
        "0" :
        {
            "x" : -4.957,
            "y" : 77.991,
            "z" : 0.0
        },
        "1" :
        {
            "x" : 23.63,
            "y" : 70.22,
            "z" : 0.0
```

```
    },
    "2" :
    {
        "x" : 14.119,
        "y" : 63.198,
        "z" : 0.0
    }
},
"3" :
{
    "items" : 3,
    "0" :
    {
        "x" : 2.349,
        "y" : 7.412,
        "z" : 0.0
    },
    "1" :
    {
        "x" : 23.725,
        "y" : 9.736,
        "z" : 0.0
    },
    "2" :
    {
        "x" : 26.344,
        "y" : -6.002,
        "z" : 0.0
    }
},
"4" :
{
    "items" : 3,
    "0" :
    {
        "x" : 8.87,
        "y" : 70.821,
        "z" : 0.0
    },
    "1" :
    {
        "x" : 5.269,
        "y" : 65.618,
        "z" : 0.0
    },
    "2" :
    {
        "x" : 37.71,
        "y" : 68.453,
        "z" : 0.0
```

```
                }
            },
            "5" :
            {
                "items" : 3,
                "0" :
                {
                    "x" : 25.635,
                    "y" : 18.969,
                    "z" : 0.0
                },
                "1" :
                {
                    "x" : 22.292,
                    "y" : 20.045,
                    "z" : 0.0
                },
                "2" :
                {
                    "x" : 20.008,
                    "y" : 18.703,
                    "z" : 0.0
                }
            }
        },
        "polygonal" :
        {
            "items" : 0
        }
    }
},
"radiating plane" :
{
    "4" : "slots",
    "l" : 63.129,
    "w" : 72.435,
    "x" : 6.473,
    "y" : -0.768,
    "slots" :
    {
        "elliptical" :
        {
            "items" : 0
        },
        "rectangular" :
        {
            "items" : 6,
            "0" :
            {
                "x" : 63.089,
```

```
            "y" : 72.31,
            "z" : 1.6,
            "l" : 4.675,
            "w" : 5.607
        },
        "1" :
        {
            "x" : 57.978,
            "y" : 81.795,
            "z" : 1.6,
            "l" : 5.223,
            "w" : 5.681
        },
        "2" :
        {
            "x" : 19.941,
            "y" : 54.015,
            "z" : 1.6,
            "l" : 4.616,
            "w" : 1.526
        },
        "3" :
        {
            "x" : 52.943,
            "y" : 19.546,
            "z" : 1.6,
            "l" : 3.536,
            "w" : 2.811
        },
        "4" :
        {
            "x" : 33.638,
            "y" : 66.181,
            "z" : 1.6,
            "l" : 2.455,
            "w" : 2.639
        },
        "5" :
        {
            "x" : 21.3,
            "y" : 47.863,
            "z" : 1.6,
            "l" : 1.687,
            "w" : 2.458
        }
    },
    "triangular" :
    {
        "items" : 2,
        "0" :
```

```
            {
                "items" : 3,
                "0" :
                {
                    "x" : 33.641,
                    "y" : -2.988,
                    "z" : 1.6
                },
                "1" :
                {
                    "x" : 23.376,
                    "y" : 3.803,
                    "z" : 1.6
                },
                "2" :
                {
                    "x" : 29.207,
                    "y" : 1.324,
                    "z" : 1.6
                }
            },
            "1" :
            {
                "items" : 3,
                "0" :
                {
                    "x" : 25.87,
                    "y" : 63.703,
                    "z" : 1.6
                },
                "1" :
                {
                    "x" : 24.153,
                    "y" : 65.628,
                    "z" : 1.6
                },
                "2" :
                {
                    "x" : 17.323,
                    "y" : 68.336,
                    "z" : 1.6
                }
            }
        },
        "polygonal" : {
            "items" : 0
        }
    }
},
"substrate" :
```

```
    {
        "l" : 83.194,
        "w" : 77.487,
        "h" : 1.6,
        "x" : -4.291,
        "y" : -3.725,
        "permitivitty" : 4.4,
        "loss" : 0.02,
        "name" : "FR4"
    }
}
```

Part 4. Technical Documentation

The printed report is supplemented with additional material that appears on the electronic medium (CD, DVD or flash disk) that accompanies this report.

Included in the additional material is technical documentation that provides more detail on the implementation, including complete software program listings, circuit designs and calculations, a user guide and all data that was measured.

The electronic medium is organised into the following directories:

1. Main report
2. Technical documentation
3. Software
4. References
5. Datasheets
6. Author
7. Datasets/Raw
8. Datasets/Final
9. Project photos